Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FIN DE CARRERA

TÍTULO DEL TFG: Visualización de datos de red

TITULACIÓN: Ingeniería Técnica de Telecomunicación, especialidad Telemática

AUTOR: Andrés Lucas Enciso

DIRECTOR: David Rincón Rivera

FECHA: 29 de septiembre de 2015

Título: Visualización de datos de red

Autor: Andrés Lucas Enciso

Director: David Rincón Rivera

Data: 29 de septiembre del 2015

Resumen

La administración de redes de telecomunicaciones es una de las actividades críticas realizadas por proveedores de red y sus operadores. Nuevas herramientas visuales permiten el diseño, planificación y administración y estudio de la evolución temporal del comportamiento de la red de forma cómoda, intuitiva y sencilla. El objetivo de este trabajo es aportar contribuciones al diseño de dichas aplicaciones de visualización.

Este documento recoge, en primer lugar, un estudio de diferentes soluciones gráficas existentes. También clasifica y compara sus modelos de visualización reflejando las principales ventajas e inconvenientes de cada uno. En segundo lugar realiza un estudio de las tecnologías actuales que supondrían una alternativa que proporcionará la integración de los aspectos positivos de cada uno de estos modelos. En tercer y último lugar, propone una solución que refleje los modelos recogidos y describa los diferentes problemas que se presentan a la hora de su realización.

Para el desarrollo de este proyecto es necesaria la obtención de datos que permitan su futura visualización. Por este motivo introduce una propuesta de estructuración de datos necesarios para su desarrollo así como un sistema de almacenamiento y una herramienta encargada de su distribución. También ha sido indispensable el desarrollo de una herramienta encargada de la creación e inyección de estos datos periódicos, imprescindible para ofrecer una sensación de interactividad al usuario final.

El objetivo final, por lo tanto, es también el diseño y la realización de una aplicación que contenga los requisitos realizados en el estudio previo y que permita la manipulación, edición y visualización de datos de red según los diferentes modelos de estudio a través de tecnologías modernas, gratuitas y de software libre.

Por último, el documento incluye un compendio de las dificultades encontradas durante el desarrollo de la aplicación y las pruebas y mejoras implementadas finalizando en la apertura de nuevas vías futuras de ampliación, revisión y mejora de la aplicación.

Title: Visualization of network data

Author: Andrés Lucas Enciso

Director: David Rincón Rivera

Date: September 29th 2015

Overview

The management of telecommunication networks is one of the critical activities made by network providers and their operators. New visualization tools allow the design, planning and administration and study of the time evolution of network behaviour in a user-friendly, intuitive and simple way. The aim of this project is to contribute to the design of the visualization tools aforementioned.

This document includes, first of all, a study of the different existing graphical solutions. It classifies and compares its visualization models in order to present the main advantages and disadvantages of each one of them. Secondly, the document contains a study of the current technologies that could represent an alternative that comprises the positive aspects of each model. Finally, the document proposes a solution that presents the models studied and describes the different problems faced during its development.

For the development of this project, the collection of valid data for their future visualization is necessary. For this reason, the solution proposes a data structure necessary for its development, as well as a storage system and a distribution engine. The development of a tool that copes with the creation and introduction of these periodical data was essential in order to offer the final-user an interactive feeling.

Therefore, the final aim is the design and development of an application that fulfils the previously studied requisites and allows the management, edition and visualization of data regarding the previously gathered models through free, open-source and modern tools.

Finally, this document includes a compendium of the difficulties found during the development of the application as well as the run-through scenarios tested and improvements implemented. The conclusion includes the guidelines for revision, expansion and future improvement of the application.

Agradecimientos

A David Rincón, mi tutor, por el tiempo invertido en este trabajo.

A mi padre, por no perder la paciencia y confiar en mí durante todo este tiempo.

A mi madre, por su perseverancia y entrega permanente.

A mi familia, por el ánimo y el cariño recibido y por aguantar mis incómodos horarios. Sin ellos no hubiese sido posible.

A mis abuelos, Pedro e Ismael, de ellos aprendí que el camino más corto no tiene por qué ser el que ofrezca mejor resultado.

A mi tío Paco, por enseñarme a no cesar en mi empeño a la hora conseguir las metas.

ÍNDICE

INTRODUCCIÓN

Vivimos en la era de la información. Internet, la red de redes, ha abierto la posibilidad de hacer llegar cualquier tipo de mensaje instantáneamente al lugar deseado con todas las implicaciones que esto comporta. Han aparecido nuevos conceptos desconocidos e inimaginables apenas hace unos años: Big Data, Internet of Things, Smart cites, etc. Los datos están cobrando una importancia mayor de la esperada y nuestra capacidad por asimilarlos no sigue este progreso a la misma velocidad.

El crecimiento de elementos con conectividad a la red y la evolución de los existentes ha creado la necesidad de mejorar y ampliar la conectividad de la red para atender esta demanda exponencial.

Este contexto aparecen abre un amplio panorama a la hora de proporcionar soluciones que faciliten la asimilación cómoda de toda esta información. La ingeniería de tráfico trata de resolver estas necesidades y mejorar los tiempos de reacción a los fenómenos de encaminamiento, dimensionado de red, balanceo de carga, recuperación de fallos, problemas de congestión, etc.

Generalmente, este objetivo ha evolucionado hacia la transformación de estos datos en elementos visuales. La traducción e interpretación de datos en imágenes se debe al uso de nuestro sentido más desarrollado. A través de la vista las personas somos capaces de comprender la información de nuestro entorno y de forma natural y rápida.

Esta memoria contiene algunas de las distintas soluciones que la ingeniería de visualización de datos de red ha desarrollado y mejorado hasta la actualidad.

Nuestro objetivo es aportar nuevas soluciones gráficas e interactivas al área de los visualizadores de datos de red. Realizaremos un proceso de inmersión en los distintos modelos que se utilizan y los importaremos al campo de nuestra investigación.

Durante la fase inicial del proyecto pretendíamos desarrollar una plataforma web de visualización vinculada al proyecto NEMO[47]. Debíamos reproducir sobre la aplicación existente otros datos complementarios (trafico, encaminamiento, flujos, etc.). Debido a limitaciones de tiempo nuestro visualizador se desligó del proyecto mencionado.

Nos centraremos en desarrollar una aplicación web que permita la visualización de estos datos utilizando todos los diseños que supongan una mejora a los actuales, mejorando así la experiencia de usuario del administrador de la red. Nuestro objetivo es crear una herramienta multiplataforma, potente y novedosa usando tecnologías modernas que supongan un salto evolutivo diferencial mediante las técnicas de desarrollo web más avanzadas, incluyendo tanto la vertiente tecnológica como la procedimental.

El documento se estructura en seis capítulos.

En el primer capítulo trataremos de introducir al lector en el contexto del proyecto. En él se pretende consensuar algunos de los conceptos básicos para la comprensión de la memoria. Incluiremos también un análisis sobre la situación de las soluciones visuales que proponen las herramientas existentes en el sector.
En el segundo capitulo introducimos los distintos modelos de visualización que existen, su funcionalidad, sus aplicaciones y las ventajas e inconvenientes que pueden ofrecer.

El tercer capitulo recoge aquellos requisitos que debe cumplir nuestra aplicación. También alberga las distintas librerías y recursos que utilizaremos para su creación.

El cuarto y quinto capítulo contienen todos los detalles acontecidos durante el desarrollo de la aplicación. Hemos considerado oportuno estructurar estos apartados según la fórmula discursiva natural para proporcionar una comprensión más agradable.

Incluimos en el sexto capítulo las conclusiones finales y las posibles líneas futuras de desarrollo.

CAPITULO 1. CONTEXTUALIZACIÓN

1.1. Presentación

Este capítulo contiene, en primer lugar, la terminología necesaria para la comprensión del proyecto. En segundo lugar, incluye la comparativa entre las tecnologías utilizadas en el sector.

1.2. Las variables

Estas definiciones nos conducirán a la asimilación común de cada uno de los conceptos durante la lectura de esta memoria. Así, evitamos las interpretaciones ambiguas que pudieran suceder.

Nodo	Cualquier elemento conectado a la red con capacidad de comunicarse con los distintos dispositivos de la misma.
Enlace	Cualquier elemento de interconexión entre distintos nodos.
Tráfico	Volumen de datos transportados entre dos nodos durante un intervalo de tiempo delimitado.
Ancho de banda	Medida de datos máxima disponible.
Congestión	Fenómeno ocurrido cuando a algún sector de una red se le ofrece más trafico que el que puede cursar.
Flujo	Corresponde al conjunto de paquetes que coinciden en la quíntupla compuesta por: dirección IP de origen, dirección IP de destino, puesto de origen, puerto de destino y protocolo de transporte. Esto se correspondería con el flujo generado por una aplicación particular, por ejemplo, una sesión web sobre el puerto 80. Se puede definir también a diferentes niveles de agregación (varias IPs de origen, diversos puertos, etc.).
Trayectoria	Conjunto ordenado de nodos que atraviesa un flujo desde el nodo de entrada hasta el de salida.
Throughput	Volumen de información neto que fluye por la red.
Sistema Autónomo	Grupo de redes IP que poseen una política de rutas propia e independiente.
Matriz de tráfico	Matriz que especifica el volumen de tráfico transportado durante un periodo de tiempo acotado entre dos nodos (origen y destino).

NREN Red nacional de investigación y educación.

1.3. Marco actual y herramientas existentes

En la actualidad existe gran demanda de calidad en los servicios de telecomunicaciones debido al progreso experimentado en los últimos años. Durante este reciente periodo, el mercado ha evolucionado desde un modelo centrado en la telefonía fija y celular hacia nuevas arquitecturas e infraestructuras de provisión de infinidad de servicios.

Esta nueva orientación requiere de un mayor ancho de banda, rendimiento y garantía en el servicio. Para poder prestar un servicio de calidad y sin interrupciones son fundamentales unas herramientas adecuadas que faciliten la buena gestión de red y el control de tráfico que circule por ésta.

Las herramientas utilizadas hasta la fecha están siendo desplazadas por otras más novedosas que ofrecen mayor oferta de servicios pero con la misma garantía y fiabilidad de resultado.

Debido a limitaciones de espacio, el anexo X recoge con detalle las diferentes herramientas analizadas en el proceso de documentación previa a esta memoria.

CAPITULO 2. MODELOS DE VISUALIZACIÓN

2.1. Introducción

La comunicación visual de datos ha existido de múltiples formas hace miles de años. La mayoría de los métodos más extendidos –line, bar y pie chart- siguen dominando las salas de juntas de infinidad de corporaciones desde el siglo XVIII. Lo que sí es nuevo es el interés contemporáneo por un tema que ha surgido de manera accidental en la última década.

El economista jefe de Google comentaba en enero de 2009 en una entrevista[1] para McKinsey&Company: “La capacidad de tomar datos –entenderlos, procesarlos, extraer su información y comunicarla – va a convertirse en una habilidad muy importante en las próximas décadas”.

Para definir este concepto resulta interesante, previamente, tener en cuenta los diferentes agentes relacionados con el intercambio de información: emisor, receptor, mensaje y canal. En el contexto de este trabajo a un lado tenemos el emisor procurando comunicar resultados, análisis o interpretaciones. Al lado contrario tenemos el receptor, público a quien va dirigida la información. El canal de comunicación es el medio mediante el que se transmite la información (un gráfico, una infografía editorial, una pantalla táctil o en este caso, una visualización web interactiva). Por último, el mensaje es aquella información a comunicar.

El principal escollo para la comunicación del mensaje es asegurar con certeza que el mensaje se comunica de la forma más efectiva y eficiente y que satisfaga las necesidades del receptor. Para ello hay que tener en cuenta las capacidades de percepción visuales del receptor según múltiples interpretaciones subjetivas eligiendo el canal más adecuado.

El proceso de comprensión de los datos comienza con un conjunto de datos y una voluntad o inquietud por comprender estos mismos. Benjamin Fry, en “*Computational Information Design*”[8] clasifica los pasos a lo largo de este proceso en:

1. Adquisición. El proceso de obtención de los datos, ya sea desde un archivo en un disco, una web, una base de datos, etc.
2. Análisis. Elaboración de un prototipo de estructura que encapsule estos datos y los ordene en categorías.
3. Filtrado. Eliminación de aquello que no merezca interés.
4. Procesamiento. Aplicación de métodos estadísticos para identificar patrones o substraerlos y contextualizarlos matemáticamente.
5. Representación. Elaboración de una representación simple de fácil comprensión como prototipo.
6. Perfeccionamiento. Mejoras en la representación con tal de hacerlo más claro y atractivo visualmente.

7. Interactividad. Adición de métodos para la manipulación de datos o el control de las características visibles. En este apartado también se incluyen a menudo las transiciones, con el objetivo de mejorar la experiencia del usuario.

Andy Kirk clasifica el proceso de diseño de visualización de datos en ocho diferentes perfiles de intervención y 5 distintos procesos de producción en la publicación "*The 8 hats of data visualization design*"[2] tal y como refleja la figura 1.

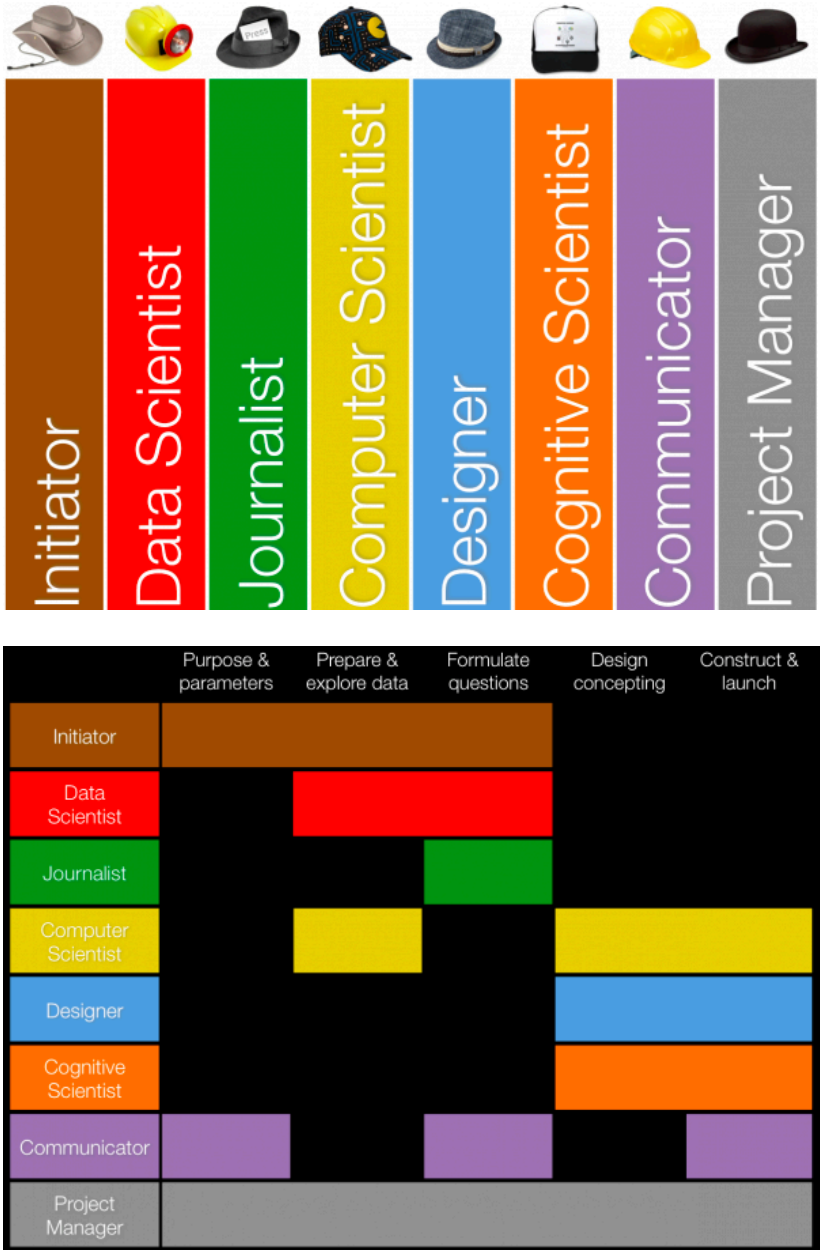


Figura 1 – Los ocho sombreros en el proceso de diseño de visualización de datos. (Andy Kirk©)

El libro "Data Visualization: A Sucessfull Design process, A Sucessful Design Process"[3], sirve de guía rápida pero integral de los mejores enfoques de diseño de visualización de datos utilizando ejemplos reales y diagramas

ilustrativos. Su lectura no requiere de conocimientos previos en la materia pudiendo satisfacer las expectativas de un amplio abanico de público.

El hecho de utilizar un canal sensorial, por el contrario, tiene como inconveniente la fácil manipulación de los resultados mediante diferentes prácticas que utilicen las deficiencias cognitivas del receptor de manera deliberada (o no). Darrel Duff, en el best seller "*How to lie with statistics*" explica diferentes técnicas de difusión de información gráfica manipulada intencionadamente en beneficio propio[6]. Queda como responsable de la codificación la ética y buena práctica del emisor.

2.2. La visualización de datos

La visualización es, literalmente, un proceso de manipulación y presentación de la información a efectos visuales codificándola y elaborando unas reglas de interpretación de los datos y expresando sus valores con propiedades sensoriales para facilitar la cómoda asimilación del receptor. La elección de la vista como principal canal para la asimilación de la interpretación de los datos es la consecuencia del establecimiento de este sentido como el más desarrollado por la especie humana y utiliza todas sus capacidades para la codificación de información.

2.2.1. Los vectores

El sentido de la vista aporta al ser humano dos sistemas de asimilación del entorno:

- El sentido de la luminosidad y el color. (Visión tri-cromática)
- El sentido espacial y de profundidad (Visión multi-ocular).

Ambas variables responden a propiedades subjetivas del receptor y, por lo tanto, están condicionadas por éste, asumiendo que en determinados casos no existirá comunicación posible según la codificación utilizada.

2.2.1.1. El vector color

El ser humano tiene una visión tri-cromática. Eso significa que es capaz de recibir la luz correspondiente a las longitudes de onda de los colores primarios (Rojo, verde y azul), combinarlos (generando toda la gama cromática perceptible) y transformar esa información en impulsos eléctricos interpretables mediante actividad cerebral.

A pesar de toda esta vertiente fisiológica el ser humano responde instintivamente a los colores debido a su significado emocional. La cromatología es el estudio del color como uno de los constituyentes fundamentales de la forma de su significante icónico. Es muy importante tener en cuenta este factor a la hora de elegir una paleta adecuada con tal de no cometer incongruencias comunicativas.

En nuestro caso, tras varios experimentos, se decidió utilizar la paleta de la figura 2.

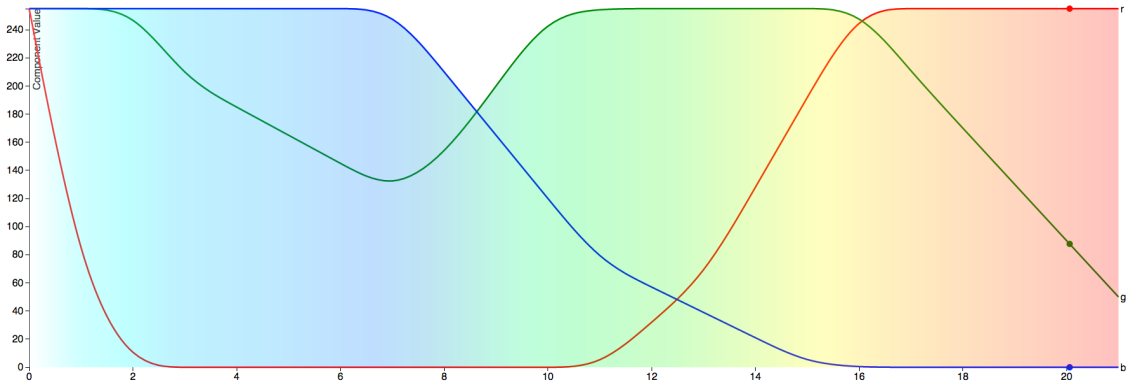


Figura 2 – Paleta de colores utilizada en el desarrollo de la aplicación.

Esta paleta refleja de forma deliberada las siguientes condiciones:

- 0% – blanco: información nula o irrelevante. Describe pureza, vacío.
- 25% – cian: información poco relevante. describe tranquilidad, armonía.
- 50% – turquesa: información sensible a considerarse. Describe naturalidad.
- 75% – amarillo: información a tener en cuenta. Describe alerta.
- 90% – rojo: información muy importante. Describe peligro.

Stephen Few, en su libro “Show me the numbers: Designing Tables and Graphs to Enlighten”[4] explica cómo la elección de unos colores adecuados pueden marcar la diferencia dedicando unos minutos a este debate durante su presentación en el SLDS Annual Grantee Meeting en Noviembre de 2008[5].

En el anexo 9.3 se detalla de manera más desarrollada las características de este vector.

2.2.1.2. El vector espacial

Existen muchos y muy diferentes modelos para la visualización de datos. A pesar de todo, la repartición de éstos puede representarse de muy diferentes formas a lo largo y ancho del espacio mediante su encolamiento, superposición, agrupación o separación de los elementos, tal y como indica la figura 3. En ella, se representan los valores correspondientes al valor de las acciones de cuatro compañías del sector de las tecnologías. La elección de cada una de estas formas de presentación depende de la información a destacar.

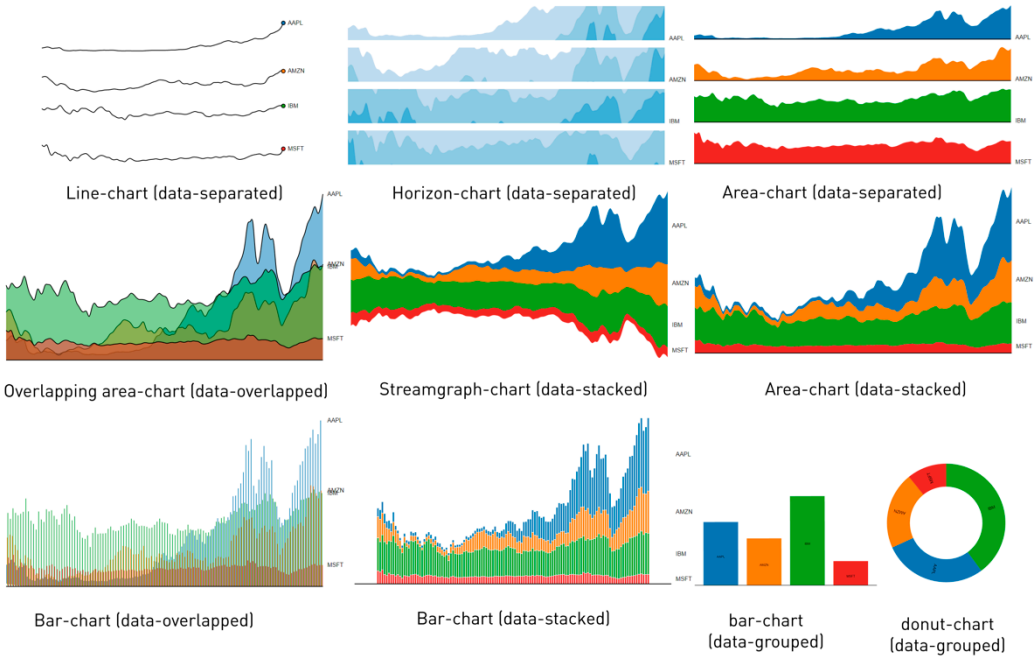


Figura 3 – Diferenciación de disposición de elementos en el espacio para la misma representación de datos[10].

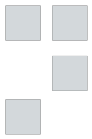
2.2.1.3. Otros

Existen infinidad de posibilidades para representar datos. Además de los descritos con anterioridad cabe mencionar, también, el uso de la textura y la forma, entre muchas otras variables capaces de realizar esta función.

2.3. Tipos de datos

Existen tres tipos básicos de datos: contables, ordenables y categóricos[9].

- Contables o cuantitativos. Cualquier unidad contable. Esta categoría incluye tanto si son valores continuos como si no. Incluimos cantidades, temperaturas, longitudes, fechas, etc.
- Cualitativos u ordenables. Cualquier dato que puede ser comparado y ordenado. Corresponde generalmente a escalas poco precisas que a menudo responden a limitaciones idiomáticas de concreción. (Todo, mucho, bastante, poco, nada || bueno, regular, malo). Un ejemplo serían los fallos en seguridad: crítico, drástico, importante, normal, superfluo.
- Categóricos. Cualquier otro dato. Corresponde con un grupo contable finito y conocido, normalmente no muy grande, y no ordenable numéricamente. Podrían ser grupos semánticos, variables booleanas, etc.

Tipos de datos	Orientación	Forma	Tono del color	Tamaño	Valor del color	Textura	Posición		
							x	y	z
Contables	X			X	X		X	X	X
Cualitativos	X			X	X		X	X	X
Categóricos	X	X	X			X	X	X	X
									

2.4. Los modelos

Existe un amplio abanico de plantillas o “layouts” de visualización de datos. Muchos de estos son muy simples de comprender debido a su popularidad. Otros, algo más complicados, se utilizan frecuentemente en entornos más específicos. En estos casos, las plantillas se generan a partir de conjuntos de datos y reglas de asignación más complejas.

En nuestra era de la información generalmente tenemos la sensación que ésta excede nuestra capacidad de procesamiento. La cantidad de información que nos rodea resulta abrumadora.

La elección de un adecuado modelo de presentación de los resultados es determinante. Por ese motivo, es imprescindible conocer las alternativas que disponemos según sus características y aportaciones.

2.5. Clasificación, dimensiones y variables

Es muy importante diferenciar entre estos dos elementos entre nuestros datos antes de elegir el modelo que deseamos utilizar. Las variables de nuestra visualización serán aquellos elementos cuyas propiedades se presentarán según sus características (dimensiones).

2.5.1. Sin dimensiones

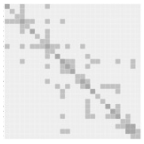
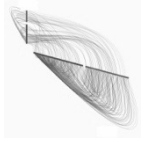
Según esta clasificación, puede suceder que nos encontremos con datos que no dispongan de ninguna dimensión o característica contable o cuantitativa ni ordenable. De esta manera, la única forma de poder presentar los resultados será su aparición en valor relativo a lo largo de la serie y presentarlos comparativamente.

2.5.1.1. Representación comparativa

Los principales modelos para la representación adimensional de datos de manera comparativa son los gráficos de barras (disposición lineal), los diagramas en donut, tarta o circular o sus múltiples variaciones geométricas.

2.5.1.2. Representación de relaciones entre conceptos

Los principales métodos son los que describe la tabla siguiente:

Topología	Arc-chart	Mat-Link	Hiveplot	Chord	Paral·lel coordinates	Scatter-plot
						

2.5.2. Unidimensionales

Corresponde a la mayoría de las visualizaciones clásicas o tradicionales. Se encarga de relacionar conceptos que disponen de una única característica con correspondencia cuantificada u ordenable con otra u otras categóricas.

2.5.2.1. Representación de series temporales continuas

Una de las representaciones más comunes es la representación de distintas variables a lo largo del tiempo acotando el inicio y el final de este. Para ello se utilizan gráficos lineales, de barras, de área, etc. Es muy frecuente encontrar el eje del tiempo orientado horizontalmente.

2.5.2.2. Representación de jerarquías

Otro de los ejemplos de utilización práctica de visualizadores es en la representación de estructuras jerárquicas. Éstas consisten en la disposición de

elementos según la característica cualitativa de su posición según distintos criterios (parentesco, organizativo, empresarial, etc.) normalmente se realiza la repartición mediante una estructura de nodos y enlaces aunque su disposición puede ser diferente tal y como se muestra en la figura 4.

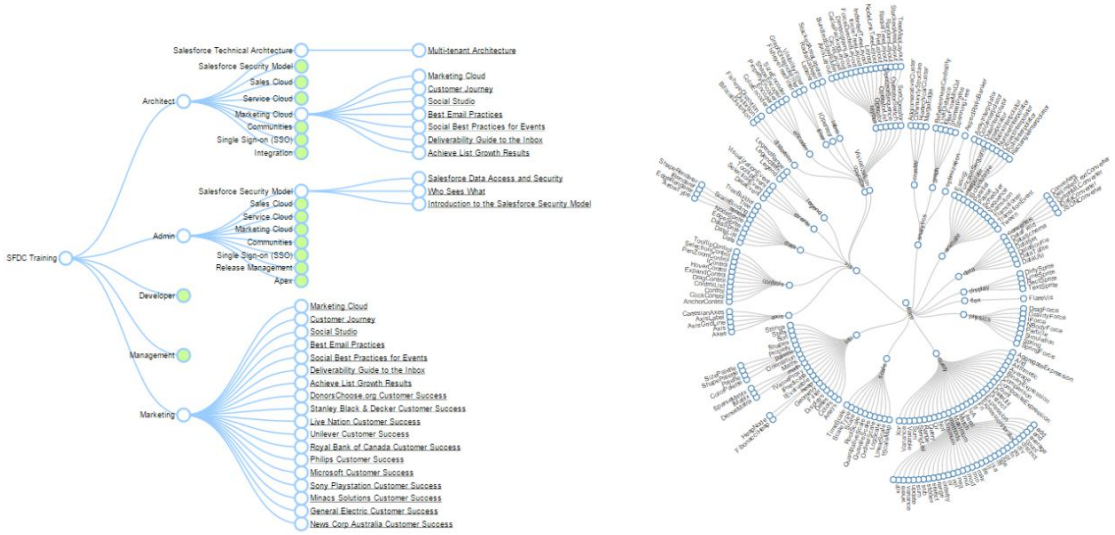


Figura 4 – Modelos de representación jerárquica con estructura de nodos y enlaces[11,12].

Existen muchas alternativas a este modelo, no tan conocidas, entre las que destacan:

- Sunburst: Disposición circular del centro hacia el exterior. Cada circunferencia circunscrita exterior representa un nivel inferior en la jerarquía. Los elementos hijos deben repartir el arco del padre. Tiene a su favor la aportación cuantificada de valor a cada nodo además de su nivel jerárquico.
- TreeAreaChart: Los elementos están dispuestos dentro del área de alguna forma geométrica. Se agrupan por colores según sus características.

Figura 5 – TreeAreaChart (izquierda) y Sunburst (derecha) son alternativas de visualización de estructuras jerárquicas[13,14].

2.5.3. Bidimensionales

Corresponde a aquellas representaciones con dos variables cuantificables y ordenables. Existen numerosas unidades de medida de magnitudes vectoriales que por características intrínsecas ya se consideran bidireccionales y cubren ambas dimensiones, un ejemplo son las geo-localizaciones, pero también existen otras como la superficie.

2.5.3.1. Representación topográfica

Las principales visualizaciones vectoriales bi-dimensionales se corresponden con localizaciones geográficas de elementos. Éstas consisten en la disposición de los elementos en el espacio x-y. En el caso de contener una geo-localización, los elementos se cargarán sobre un mapa con la proyección correspondiente.

2.5.4. Multidimensionales

Las visualizaciones multidimensionales atienden a datos de más de dos variables cuantificables u ordenables. Su representación no tiene por qué tener una interpretación complicada.

Un ejemplo de modelo de visualización sencilla de este tipo es el denominado Radar chart o Spider Chart

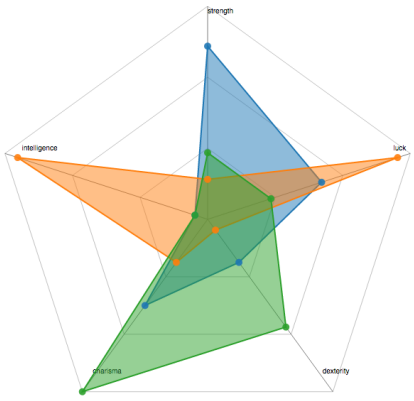


Figura 6 – Spider/Radar chart[15].

2.5.4.1. Otras visualizaciones complejas

Existen otros modelos complejos que son capaces de representar múltiples magnitudes cuantificables u ordenables además de poder establecer relaciones entre ellas. Estas visualizaciones, en cualquier caso, requieren de un periodo de adaptación mayor debido a su complicada asimilación y generalmente se utilizan para presentar información sin demasiada precisión individual resultando muy interesantes para conseguir asimilar rápidamente la idea general[18]. Unos ejemplos pueden ser Circos[16] y Hive-plot[17].

2.6. Análisis visual efectivo

La codificación efectiva mediante visualizadores debe seguir la misma directriz de diseño resumida en una simple norma: "*Overview first", zoom and filter, then details-on-demmand*"[7].

Tener una visión general es muy importante. Reduce la búsqueda, permite y facilita la detección de patrones y ayuda al receptor en la elección de la siguiente acción. Una práctica acertada es comenzar con una visualización general del diseño. También es necesario, además, permitir al usuario el acceso a la información que desea rápidamente.

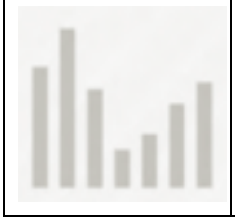
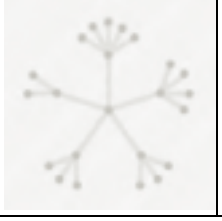
Una solución es proporcionar una visualización con zoom y permitir múltiples enfoques una visión general y una visualización muy precisa para futuros análisis.

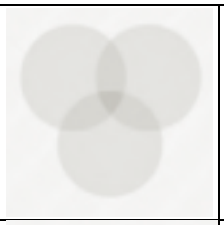
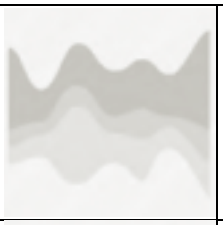
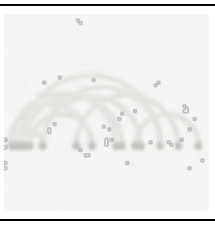
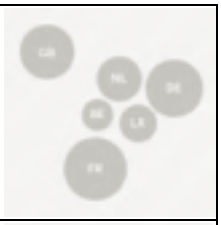
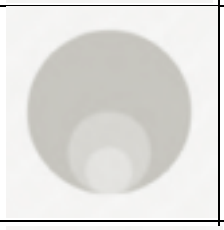
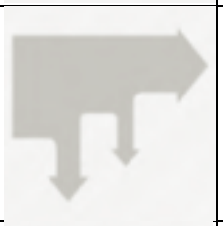
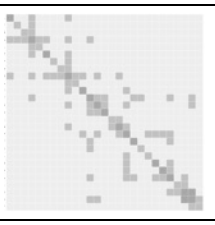
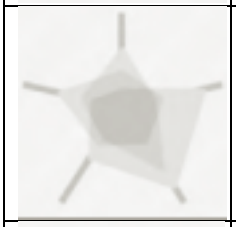
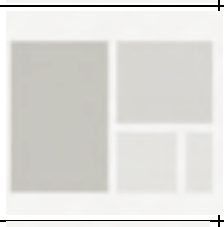
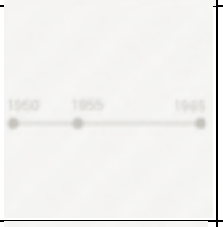
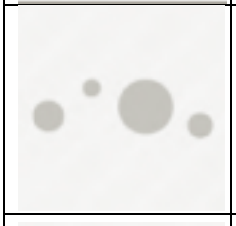
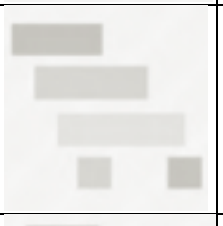
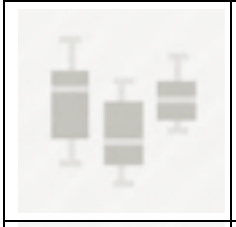
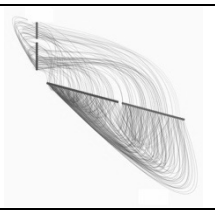
El receptor también utiliza muy frecuentemente la decisión de omitir (filtrar) aquella información por la que no está interesado reduciendo así el conjunto de datos presentados.

Después de algunas acciones el usuario acaba obteniendo la selección de sus datos de interés. Este proceso iterativo de refinamiento o consulta progresiva de los datos es lo que se conoce como la toma de decisiones jerárquica (hierarchical decisión-making).

2.7. Resumen de los modelos según su finalidad

Sirva la siguiente tabla como resumen de algunos de los principales modelos de visualización en función del objetivo teniendo en cuenta todos los aspectos detallados previamente.

Comparativa entre valores cuantitativos y jerárquicos	Estructuras jerárquicas	Visualización de evoluciones y cambios temporales	Visualizaciones relacionales y saltos o variaciones	Visualizaciones espaciales
				

Comparativa entre valores cuantitativos y jerárquicos	Estructuras jerárquicas	Visualización de evoluciones y cambios temporales	Visualizaciones relacionales y saltos o variaciones	Visualizaciones espaciales
				
				
				
				
				
				
				

CAPITULO 3. PROPUESTA DE SOLUCIÓN

3.1. Introducción

El abanico de herramientas existente en la actualidad para la visualización de datos es muy amplio y es fundamental la elección de la herramienta adecuada para este propósito, previo a abordar la fase de producción.

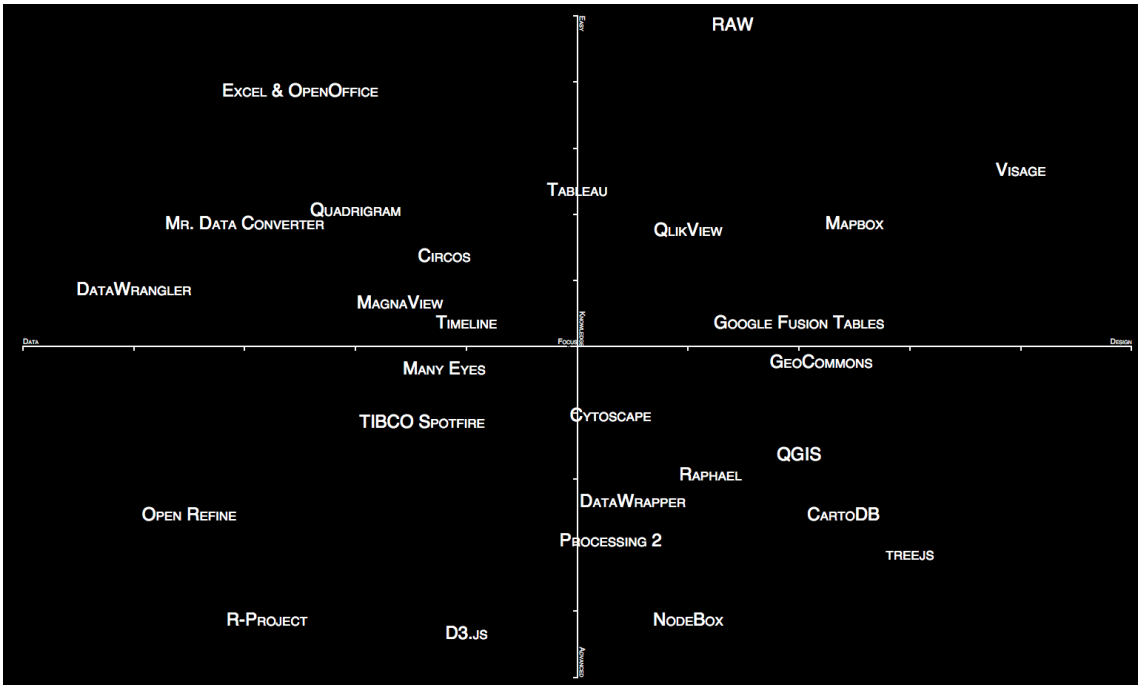


Figura 7 – Herramientas de visualización de datos dispuestas sobre un plano cuyo eje vertical refleja la complejidad (fácil - complicado) y el horizontal el principal interés (datos - diseño)[19].

Partiendo de la premisa de obtener una tecnología que permita perdurar el mayor número de años vigente, debemos entender que ésta deba utilizar las últimas y más novedosas técnicas de desarrollo sin perder de vista la penetración en el mercado a la vez de atender a las nuevas fórmulas de diseño de interfaces gráficas para el manejo cómodo y comprensivo del usuario.

3.1.1. ¿Por qué interactiva?

Las visualizaciones estáticas pueden ofrecer visiones pre-concebidas de datos, por lo que a menudo es necesario un gran número para presentar infinitas perspectivas de la misma información. Cuando los elementos visuales intervienen en la misma plataforma en un mismo momento, el número de dimensiones de datos es limitado también. La representación de conjuntos de datos multidimensionales de manera precisa en imágenes estáticas es notoriamente complicada. La imagen estática es ideal cuando el resto de

alternativas no son deseadas o necesarias, como, por ejemplo, cuando realizamos una publicación impresa.

Las visualizaciones dinámicas interactivas predisponen a la gente a observar la información por sí mismos siguiendo el criterio organizado de presentación "*Overview first", zoom and filter, then details-on-demmand*"[7]. La interactividad también puede alternar la visualización con los datos, cosa que las imágenes estáticas no pueden. Con transiciones animadas e interfaces bien elaboradas, algunas visualizaciones pueden hacer que los datos de la exploración se sientan más como un juego. La visualización interactiva puede ser un medio perfecto para llegar a aquellos que consideran el aprendizaje una tarea aburrida.

3.1.2. ¿Por qué en formato Web?

Las visualizaciones no son realmente "visibles" a menos que las observen. Por este motivo, Internet y la plataforma web nos ofrece la forma más rápida de llegar a una audiencia global.

Todo lo cubierto en este proyecto puede realizarse con herramientas de software libre (open-source), por lo que la única inversión necesaria es el tiempo. Todo lo utilizado es código abierto y estándares web. Al evitar software privado y otros plugins nos aseguramos que los proyectos son accesibles en el mayor número de dispositivos. Cuanta más accesibilidad en la visualización mayor será el público y su impacto.

La herramienta web es perfectamente compatible con todos los detalles especificados en la actualidad, ya que nos permite desarrollar código interactivo y visual, proveyéndonos además de las virtudes de esta plataforma (velocidad y audiencia global). Para ello, lo único que hay que asegurarse es estar utilizando las herramientas adecuadas.

3.1.3. ¿Por qué modular?

Las herramientas actuales son habitualmente muy complejas. La mente humana tiene una capacidad de racionamiento limitada y le resulta imposible abordar los problemas en conjunto. Por ese motivo seccionamos cada uno de los problemas que nos encontramos en pequeños conjuntos a los que abordar poco a poco y luego integramos todas las pequeñas soluciones para conseguir el resultado final. En el campo de la programación, este método tiene como ventaja que, teniendo en cuenta la rápida evolución de las tecnologías, supone un problema menor a la hora de reemplazar determinadas secciones que pudieran quedar desfasadas.

3.1.4. ¿Por qué segura?

La seguridad se ha vuelto un tema muy importante en Internet. La red de redes, que nació con finalidades militares, ha considerado fundamental este concepto desde sus orígenes. El protocolo web es muy extendido actualmente entre usuarios sin conocimientos técnicos y como consecuencia existe un gran interés por la substracción de información de éstos por parte de usuarios malintencionados, con mayores conocimientos técnicos. Uno de los requisitos será crear una herramienta capaz de proveer una pasarela con distintos perfiles de usuario y permisos.

3.1.5. ¿Por qué confidencial?

Por muchas medidas de seguridad que apliquemos, hemos de reconocer que desgraciadamente siempre quedará una posibilidad de ser víctima de un ataque. Al estar almacenando datos de usuario y teniendo en cuenta los principios de confidencialidad y los artículos correspondientes a la ley de protección de datos, ofuscaremos cualquier contraseña y dato sensible de ser sustraído.

3.1.6. ¿Por qué adaptable?

Vivimos en la era de “Internet of things”. Cada vez encontramos elementos nuevos o existentes con conectividad a la red global. El servicio web se ha tenido que adaptar a esta tendencia. En los últimos años hemos podido observar cambios muy importantes en los diseños de las páginas y las evoluciones de los estándares han permitido tal evolución. Infinidad de dispositivos de muy diferentes tamaños pueden cargar elementos web, por lo que es imprescindible diseñar una aplicación considerando las últimas tendencias. Los diseños “*responsive*” son capaces de cargar de manera flexible el código, adaptándose al dispositivo demandante.

Al poder cargar de una forma amigable las visualizaciones en dispositivos “*wereables*”, ofrecemos al usuario una mejor experiencia y además, le liberamos de la obligación de tener que estar delante de la computadora para realizar el seguimiento de la red.

3.1.7. ¿Por qué amigable?

Los diseños web están evolucionando hacia interfaces “*user-friendly*”. Cada vez existen más diseñadores de interfaces de usuarios y esta especialización es muy demandada en muchos sectores. Proveer la herramienta de estas corrientes mediante una interfaz simple y sencilla pero que sin renunciar a ninguna funcionalidad es un aspecto más a considerar.

3.1.8. ¿Por qué con un instalador?

El objetivo es generar una herramienta que sea fácilmente extensible. Al proporcionar un instalador en la propia herramienta mejoramos la experiencia del encargado de la instalación. Cuanto más cómodo sea el proceso de instalación mayor garantía de expansión.

3.2. Lenguajes de programación

3.2.1. Introducción

Una fase inicial de requerimientos nos ayuda a elegir códigos de programación que permitan realizar todos los aspectos anteriores. Es importante dedicar el tiempo necesario a esta fase de diseño de la arquitectura o de lo contrario, en un futuro nos podríamos encontrar sorpresas indeseables y haber trabajado en vano.

Los requisitos de esta aplicación web son:
- Visualización de datos…
  - …en tiempo real.
  - …de series temporales.
  - …interactivos.
- Almacenamiento…
  - …de datos confidenciales
  - …de datos privados
  - …de datos jerárquicos
  - …de datos cíclicos
  - …de un volumen considerable de datos.
- Aprovisionamiento de datos…
  - …bajo demanda.
- Utilización de código open-source.
- Diseño de una interfaz amigable
- Respuesta a múltiples perfiles de usuario en función de sus permisos.
- Creación de una arquitectura…
  - …modular.
  - …escalable.

Resulta interesante dividir estos requisitos en tres apartados según el dispositivo encargado de su materialización:
- La base de datos, encargada de almacenar y proporcionar los datos confidenciales y seguros bajo demanda y bajo contraseña.
- El servidor, encargado de proveer la información requerida al cliente.
- El cliente, generalmente el navegador web, encargado de procesar y presentar al usuario la información.

Tendremos en cuenta también la delegación de cualquier proceso al lado cliente con tal de ahorrar carga al servidor y mejorar su servicio.

Teniendo en cuenta todas las características anteriores proporcionamos una propuesta capaz de contener todas las características anteriormente citadas. Ésta es una multiplataforma que ofrece un sistema distribuido, escalable, de óptima seguridad, confidencial y de uso simple.

3.2.2. Base de datos.

La elección de la base de datos fue una característica muy particular ya que fue contraria al procedimiento habitual. En este caso, la limitación la daba el escenario, que solo nos proveía de un servicio de gestión de almacenamiento mediante MySQL5.5 convirtiendo esta elección en una decisión "*de facto*" en lugar de "*de iure*". Por ese motivo, el estudio se enfoca principalmente a comprobar si este sistema de gestión de datos cumple todos los requisitos. Aun así, se realiza una pequeña investigación sobre las posibilidades que ofrecían los otros sistemas de gestión que se detalla en el anexo I.

MySQL es un sistema administrador de base de datos que permite la gestión de gran cantidad de datos de manera muy efectiva. Es el más popular y extendido convirtiéndolo en una posibilidad gratuita para cualquier usuario. También existe una amplia documentación al respecto. En cuanto a los tipos de tablas escogidas, en nuestro estudio nos decantamos por InnoDB, de transacción segura a diferencia de otras también muy populares como ISAM, MyISAM, MERGE o HEAP.

La mayoría de los sitios web utilizan tablas MyISAM ya que realizan pocas consultas de tipo INSERT o UPDATE en comparación con las de tipo SELECT. En nuestra propuesta ocurre precisamente lo contrario y por lo tanto la optimización que aporta este tipo de tablas a las consultas SELECT no supone ningún interés frente a la garantía de transacción segura que ofrece InnoDB.

Existe una limitación muy importante en la elección de las tablas MySQL para la base de datos y es que no ofrecen almacenamiento mediante un modelo de Planificación Round Robin (RRD)[21].

A pesar de ello, podemos encontrar un artículo de Oli Sennhausser titulado "*Round-Robin Database Storage Engine (RRD)"* que discute sobre la posibilidad de almacenar información de este tipo en MySQL utilizando triggers[20].

3.2.3. Server side

La aplicación está preparada para servidores Linux, que son los más populares y extendidos. Otras alternativas son Windows, OS X, BSD y Solaris principalmente, pero las siguientes razones hacen de esta infraestructura (Linux) la mejor candidata:

1. *Popularidad*. Los sistemas operativos más extendidos en el mercado son Windows y Linux. Hay que mencionar que una mayor cuota de mercado garantiza mayor documentación en Internet en caso de necesidad.
2. *Estabilidad*. Los sistemas Linux son reconocidos popularmente por su característica de funcionar durante años sin fallos. Linux, además gestiona multitud de procesos concurrentes mejor que Windows. No hay necesidad de reiniciar el sistema en caso de actualización o cambio de configuraciones.
3. *Seguridad*. Linux en más seguro que Windows tanto como servidor, ordenador personal o sistema integrado. Linux, basado en Unix, fue concebido para ser un sistema multiusuario. Tan solo el administrador o super-usuario tienen privilegios para realizar cambios de configuración y usuarios y aplicaciones no tienen permisos de acceso al kernel ni entre sí. Esto permite conservar cada módulo protegido.
4. *Hardware*. Mientras que Windows requiere, a menudo, de constantes actualizaciones de hardware para soportar sus crecientes demandas de requisitos, Linux es ligero, ajustable, modular, escalable, flexible y se integra formidablemente en cualquier computadora sea cual sea su procesador o arquitectura.
5. C*oste Total de Propiedad (TCO)*. No supone ningún coste adicional ya que el software se encuentra generalmente bajo licencia libre. Inclusive una versión corporativa empresarial supondrá generalmente un coste menor que Windows o cualquier software propietario que implicará la compra de licencias de usuario y un sinfín de complementos caros, esencialmente de seguridad.
6. *Libertad*. En Linux no existirá ningún vendedor tratando de orientarnos hacia la compra de ciertos productos o el uso de determinados protocolos. Sencillamente nos proporcionará la libertad de poder escoger aquellos que mejor se adapten a nuestra tarea.

Destacamos además, dentro de las posibilidades de servidores del mercado, la mención de IIS (Inernet Information Services) para Sistemas Windows que no nos ocupa y Apache HTTP para Linux. La elección de este último es sencillamente por su popularidad, ya que existen otros más veloces y eficientes. Particularmente Ngibx, Lighttpd y LiteSpeed resultan destacables y son tecnológicamente superiores.

El lenguaje utilizado es PHP5. El propósito del proyecto pretende mostrar resultados variables. Para esto se precisa de algún lenguaje que pueda ofrecer flexibilidad en la página web cliente. El lenguaje PHP es un lenguaje de programación de código en el lado del servidor que incorpora directamente lenguaje en el documento HTML. Se considera uno de los lenguajes más flexibles, potentes y de alto rendimiento conocidos hasta la fecha. La utilización de este lenguaje en el lado servidor será de gran utilidad para trabajar con datos variables. Este lenguaje permite la utilización de objetos, es moderno, de alto nivel y muy popular, y a pesar de que existen otros muchos y más modernos (como Python o Ruby) nuestro desconocimiento técnico hizo decantarnos por la opción de menor riesgo.

No se ha probado la instalación en ningún otro sistema operativo así como tampoco ninguna otra plataforma IIS por lo que no podemos descartar su funcionamiento.

3.2.4. Client Side (Javascript)

Debido a la gran cantidad de carga que supone realizar todos los cálculos en la máquina servidora, se delegaran todas las operaciones computacionales posibles a las máquinas usuarias reduciendo así la carga de éstos. Por ese motivo y para mejorar la interfaz de usuario utilizaremos Javascript. Es un lenguaje de programación interpretado, orientado a objetos, basado en prototipos, imperativo, débilmente tipificado y dinámico. Se utiliza principalmente en el lado cliente, implementado como parte del navegador web. Todos los navegadores modernos lo interpretan.

3.2.4.1. Carga dinámica de la página

Debido a la gran cantidad de elementos e información existentes hay que considerar la posibilidad de cargar las páginas dinámicamente. Esto quiere decir que el contenido de la misma se volcará bajo demanda cuando la página esté cargada. Así mejoramos tiempos de respuesta y optimizamos la carga en ambos lados de la comunicación al realizar el proceso de filtrado de tal página.

AJAX (Asyncronous Javascript And XML) es una técnica de desarrollo web sobre Javascript para la creación de contenido dinámico (consultas síncrono y asíncrono), compatible con toda la infraestructura anterior, que nos permitirá realizar cambios sobre la página sin necesidad de recargarla y mejorar así su interactividad, velocidad y uso de las aplicaciones. AJAX es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página.

3.2.5. HTML5 y CSS

Estos dos lenguajes utilizados no se han incluido en ninguno de los apartados anteriores porque son el canal de comunicación entre cliente y servidor y no pertenecen a uno concreto. HTML5 + CSS3 son estándares web HTTP popularizados y perfectamente compatibles de soportar el proyecto. Cualquier usuario del mundo conectado a internet dispone de un navegador capaz de renderizar webs que entiendan estos protocolos.

3.2.6. JSON

Para la capa intermedia entre los datos en MySQL y el lenguaje HTTP cliente, precisamos, además de PHP para su manipulación, otro lenguaje que se encargue de tu envío de manera eficiente, práctica y sencilla. La mejor técnica

de envío para la interpretación de datos en Javascript y AJAX es el lenguaje JSON (JavaScript Object Notation). Elegimos este lenguaje por su fácil ensamblaje con el lenguaje de visualización, el Javascript.

3.3. Librerías

Con el fin de reducir el tiempo de desarrollo y facilitar la tarea decidimos utilizar una serie de librerías de cada uno de estos lenguajes.

3.3.1. PHPMailer

Líbrería PHP encargada de gestionar el envío de notificaciones vía mail de la aplicación tanto para los registros como para posibles alertas de superación de lindares.

3.3.2. PHP Logger

Librería encargada de registrar en archivos de texto los errores de ejecución de PHP en el lado servidor dejando un registro diario de estos mismos.

3.3.3. jQuery

jQuery es una biblioteca JavaScript rápida, pequeña y con un gran abanico de funcionalidades. Habilita funciones como recorrido y manipulación de elementos del documento HTML, manejo de eventos, animaciones, consultas AJAX y es mucho más sencilla, mediante una API compatible con todos los navegadores modernos. JQuery, con su combinación de versatilidad y extensibilidad ha cambiado la forma en que millones de personas escribían Javascript. Su uso ha ido de la mano de infinidad de cambios y mejoras en el diseño web siendo uno de los responsables más directos.

3.3.4. Bootstrap (CSS + Javascript)

Bootstrap es un proyecto de código abierto hospedado, desarrollado y mantenido en GitHub que proporciona un entorno rápido y sencillo de desarrollo front-end para dispositivos de cualquier resolución, proyectos de cualquier dimensión y persones de todo tipo de nivel.

El código proporciona un compendio de archivos que, una vez incluidos en el proyecto nos ofrece un entorno que escala y redimensiona eficientemente el código HTML. También ofrece una amplia documentación para los elementos más comunes de HTML, docenas de componentes CSS y HTML e impresionantes plugins jQuery. En este caso se tuvo que utilizar algunos de ellos, cómo bootstrap tags-input, bootstrap Touch-spin.

3.3.5. D3.js

Biblioteca en Javascript para la producción de visualizaciones de datos dinámicos e interactivos en los navegadores web. Sucesor del framework Protovis, hace uso de SVG, HTML5 y CSS3 y a diferencia de muchas otras librerías, D3.js permite un gran control sobre el resultado visual final.

3.3.6. Javascript Google Maps API

Se trata de un servicio gratuito actualmente sin publicidad. Mediante esta API es posible incrustar Google Maps en sitios web externos. La API Javascript fue la originaria y actualmente existen cuatro distintas: Android, iOS, Javascript y HTTP. En el caso del aplicativo, utilizamos la librería Javascript en su tercera versión adaptada para la tercera versión de Google Maps.

3.3.6.1. Javascript Google Maps API

Proyecto de código abierto que es un repositorio central de las bibliotecas de servicios públicos que se pueden utilizar con la API de Google Maps v3 JavaScript.

RichMarker
Permite crear un marker personalizado extendiendo esta clase.

MarkerClusterer
Permite crear y manipular los clusters por cada nivel de zoom del mapa en caso de la existencia de gran cantidad de markers evitando así la superposición.

InfoBox
Permite la inclusión de contenido HTML en la ventana desplegable de información del marcador del mapa así como su personalización.

3.4. Hooks y plugins

Con motivo de ahorrar esfuerzos y tiempos en la programación, es interesante utilizar algunas herramientas de uso habitual. Cabe decir que, en este caso, hemos tenido que prescindir de la premisa de realizar el trabajo a coste 0. Por el contrario los resultados obtenidos ofrecerán mayor garantía.

3.4.1. Arfaly – Powerful & responsive multi-file uploader

Arfaly[22] es un importador de archivos muy utilizado y popular. Ofrece diseño "*responsive*" y realiza las comprobaciones de los archivos en el lado servidor

para evitar el hacking mediante el reemplazo de código en la máquina cliente. Esta herramienta nos reducirá el tiempo de programación concebido para el importador de archivos de datos.

3.4.2. Premium login Authentication System

EdcoreWeb ofrece una solución completa de seguridad en la web llamada "Premium Login Authentication System"[23]. Integra todas técnicas más modernas, como autenticación por redes sociales, recuperación de contraseña, registro, envío de mails, etc. Todo ello está integrado en bajo un motor Ajax de gestión de acciones. El uso de esta herramienta nos ahorrará considerables horas de programación, dejando tan solo una labor tan escabrosa como es la seguridad del portal en un mero trabajo de ensamblaje, adaptación y personalización.

3.4.3. MySQL Conversion Class

Para facilitar la comunicación con la base de datos utilizaremos una herramienta intermediaria sobre la que implementaremos toda la API-RPC. Toolkitsystems ofrece la denominada "MySQL conversión class"[24]. Ofrece muchas posibilidades, como Importación/exportación de datos MySQL, Soporte a drivers php MySQL y MySQLi y lo que es más importante para nuestro trabajo, respuesta de valores en múltiples formatos (array, XML, JSON).

3.5. Templates

Lo interesante y prioritario en este trabajo es la visualización de datos. Por ese motivo, ahorraremos tiempo en programación y maquetación de ésta re-utilizando código.

Para el caso particular de los "*templates*", la oferta es muy extensa. Tympanus es una empresa que realiza publicaciones frecuentes bajo su sección Codrops[25] utilizando las ultimas y más novedosas técnicas de diseño web y a su vez explica y libera el código del ejemplo para el aprendizaje colectivo. De esta página es interesante ensamblar los ejemplos que se adaptan a nuestras necesidades de diseño:

- Fullscreen Form Interface[26]. Muy interesante para ofrecer una navegación práctica, cómoda y simple durante el proceso de instalación de la herramienta evitando posibles distracciones.
- Custom Login Form Styling[27]. Plantilla muy elaborada de registro y acceso a la página.
- Google Nexus Website Menu[28]. Navegación de tipo drop-down-hamburger-menu que prioriza el espacio de navegabilidad ocultando en todo momento los menús de navegación mientras no se requieran.
- Fullscreen Overlay Effects[29]. Menú de superposición global. Uso muy reducido pero de interfaz eficaz y agradable.

3.6. Estructura

El visualizador está estructurado por bloques y en su conjunto forman una solución multifuncional. Según en qué nivel lo analicemos destacaremos las siguientes clasificaciones:

Por funcionalidad: Define las distintas partes que se encargan de gestionar el visualizador

- Instalador
- Registro y verificación de usuarios
- Editor de contenido (Admin)
- Visualizador

Por estructura-anidación: el proyecto está dividido en directorios claramente diferenciados ofreciendo un modelo MVC para su sencilla y práctica manipulación. Entre estas estructuras destacamos:

- Clases: Se encuentran en el directorio 'php/api-rest'. Actúan como modelo de objetos que comunican con la base de datos.
- Ajax.php: Este archivo se encuentra en el directorio raíz y es el encargado de toda la comunicación entre el back-end y la visualización. Actúa como controlador. Mediante llamadas a ésta se realizan todas las acciones posibles. Todos los resultados se obtendrán mediante datos con estructura JSON, de manera que atendemos la idea de crear una API (Application Programming Interface)completa para, en el futuro, obtener mejor escalabilidad y migración.
- Front-end. Se encuentra en el directorio 'templates'. El archivo 'index.php' del directorio raíz es el encargado de redireccionar o incluir los archivos correspondientes según unas condiciones programadas. Se trata de la capa que interactúa con el usuario.

Por jerarquía de carpetas u organización:

- Raíz: Archivos php base sin código HTML.
    - ajax.php: se encarga de la comunicación entre los modelos y su presentación. Devuelve HTML en formato JSON para el uso y manipulación de estos datos.
    - config.php: se crea automáticamente tras la instalación. Contiene las variables php globales almacenadas una tras otra.
    - definitions.php: archivo que se encarga de incluir todas las definiciones para el correcto funcionamiento de la aplicación. Incluye las variables globales, el conector a la BBDD y el logger y Mailer php.
    - initialize.php: archivo ejecutado una vez completada la instalación que crea y configura toda la base de datos con los parámetros introducidos.
    - timer.php: patch creado para lanzar la instrucción de lanzamiento del cron de actualización cada segundo debido a que el servidor compartido únicamente permite actualizar cada 1min.
    - index.php: archivo que gestiona todo el comportamiento de la vista del modelo.

- css: carpeta donde se almacenan los archivos CSS de la aplicación
- img: carpeta donde se almacenan algunas imágenes necesarias para la aplicación
- js: carpeta donde se almacenan los javascripts.
- log: carpeta donde se almacena el registro diario de errores php de la aplicación.
- php: carpeta que almacena las clases objeto de la aplicación.
- templates: carpeta que incluye cada uno de los templates de la aplicación incluidos por "index.php". Está estructurado por subdirectorios.
    - Edit: para los templates de gestión del administrador.
    - Install: para los templates del módulo de instalación.
    - Login: almacena los templates de login, registro y otros derivados.
    - Visualizations: almacena los archivos que proporcionan las visualizaciones de datos.

También podemos dividir este proyecto según los permisos en que nos encontremos:

- Instalación: si todavía no hemos instalado el producto. Debemos disponer tan solo de una base de datos creada en un servidor y su usuario y contraseña.
- Login: Si todavía no hemos registrado nuestras credenciales el acceso a la aplicación estará cerrada.
- Usuario: Tan solo permite el acceso al contenido de visualización de datos pero no podemos modificar estos.

Administrador: podemos tanto visualizar como manipular los datos de la base de datos.

3.6.1. La base de datos

Para la base de datos elegimos la tecnología más extendida en el sector: MySQL. Dispone de una amplia documentación al respecto y ofrece todas las características necesarias para este proyecto.

La figura 8 nos nuestra las tablas de la base de datos al completo y nos ayuda a comprender cada una de las particularidades que comportan cada una de ellas y las relaciones que existen entre si a la vez que nos proporciona una idea de la disposición global de ésta misma.

En la base de datos diferenciamos cinco categorías distintas de tablas, todas ellas igual de imprescindibles:

- El registro de opciones y configuraciones de la herramienta
- El registro de usuarios del aplicativo.
- Las tablas de almacenamiento de índices de datos de tipo nodo, link y otros.
- Las tablas de almacenamiento de datos de tipo RRDB.
- La tabla de elementos del disparador de eventos.

En el Anexo 9.4 se encuentran explicadas las estructuras de cada una de estas secciones.

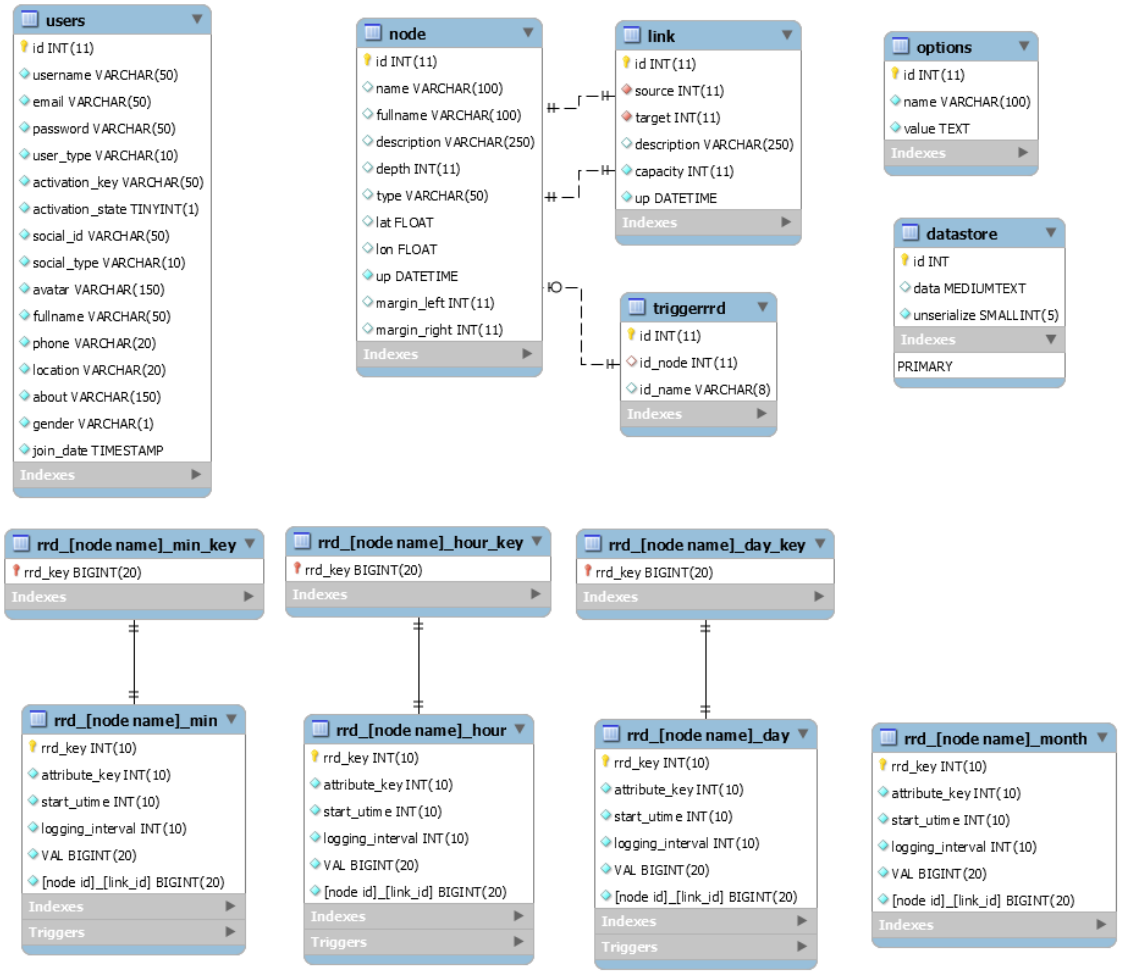


Figura 8 – Resumen conceptual del diagrama relacional y estructural de la base de datos.

3.6.1.1. Los disparadores (triggers)

Las bases de datos con estructura Round robin requieren de una serie de directrices para su funcionamiento. Éstas consisten en unos eventos que se ejecutan después de cada inserción preparando esta tabla para la siguiente y se denominan triggers.

3.6.2. El back-end

Esta sección del proyecto está desarrollada mediante código PHP. Se encarga de gestionar las sesiones de usuario, la modificación y aprovisionamiento de los datos, la simulación de éstos y la generación de las páginas según las acciones realizadas por el usuario. Los archivos siguen una inclusión lógica para ahorrar líneas de programación. Utilizamos la programación orientada a objetos para la gestión de aquellos modelos de utilización frecuente. En el apartado 3.7 de la memoria se desarrolla en profundidad este apartado.

3.6.3. El front-end

El front-end costa de todos aquellos elementos recibidos por el navegador del usuario cliente. Se trata de código HTML, CSS y Javascript (y AJAX). A modo resumen diferenciamos estos tres conceptos de la siguiente manera:

- HTML: Objeto de información (qué).
- CSS: La forma de visualizarlo (cómo).
- Javascript: Transformaciones finales (con qué cambios).
    - AJAX: Transformación demorada vinculada al objeto de información.

3.7. La lógica de funcionamiento

La figura 9 nos sirve como referencia a la hora de entender la lógica de carga de cualquier página de la aplicación.

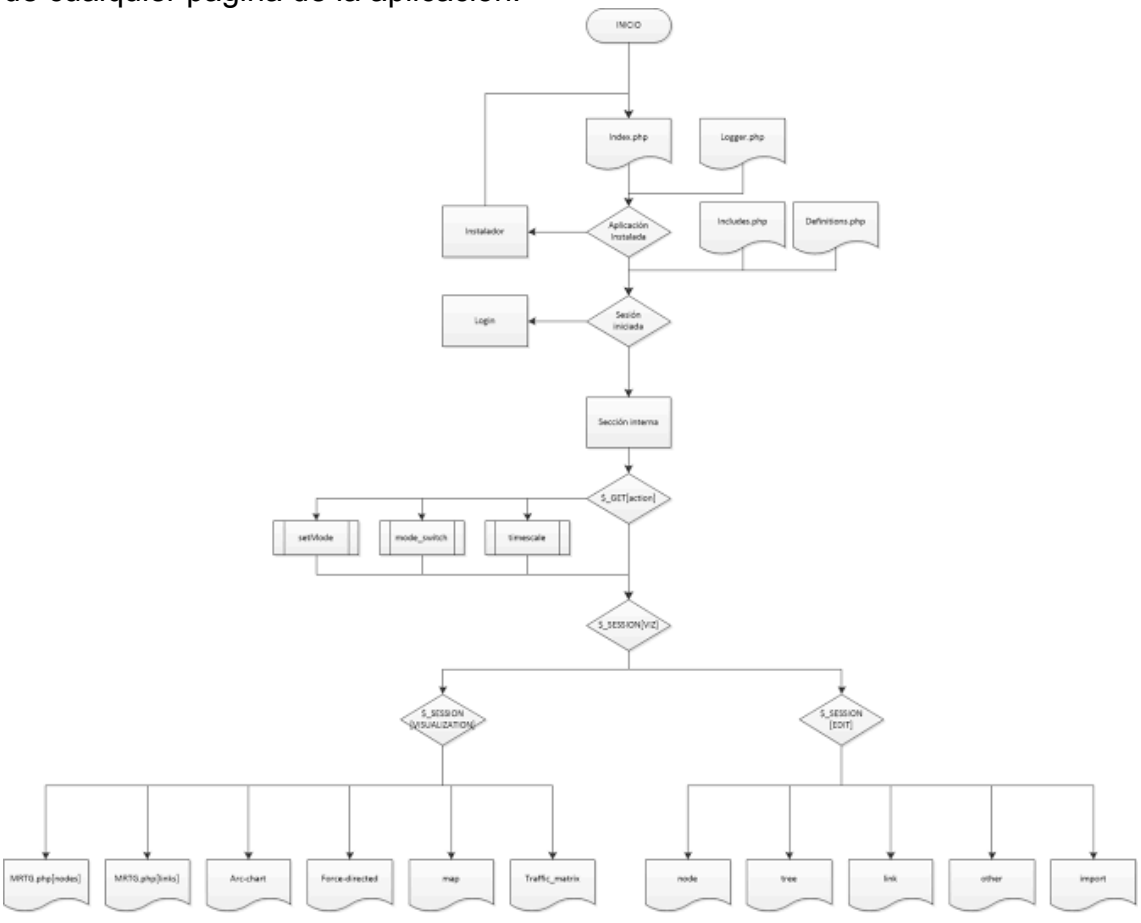


Figura 9 – Diagrama de funcionamiento de la aplicación.

3.7.1. Modelo-Vista-Controlador (MVC)

El patrón MVC fue una de las primeras ideas en el campo de las interfaces gráficas de usuario. Es un patrón de arquitectura de software que separa los datos y la lógica de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y comunicaciones. Por esto, con el fin de separar la definición de componentes para la representación de la información de la de la interacción con el usuario, MVC propone la construcción de tres componentes distintos:

- Modelo: Representación de la información por la que el sistema opera. Encargado de la gestión de todos los accesos a la información, las actualizaciones, privilegios de acceso, etc. Envía a la Vista aquella información que en cada momento se le solicita. Las peticiones de acceso o manipulación llegan al modelo a través del controlador.
- Vista: Presenta el modelo en un formato adecuado para interactuar (Interfaz de usuario o UI) y requiere del modelo la información que debe representar como salida.
- Controlador: Responde a eventos (generalmente acciones de usuario) e invoca peticiones al modelo al realizar solicitudes de información. También puede enviar comando a su vista asociada si se solicita un cambio en la forma en que se presenta el modelo. En general ejerce de intermediario entre vista y modelo.

3.7.2. El Modelo

El modelo se encuentra incluido en la cabecera de los archivos principales de funcionamiento de la aplicación, como son ‘index.php’ y ‘ajax.php’ del directorio principal. Cada una de las clases incluidas es la encargada de proporcionar los datos correspondientes a sendas tablas en la base de datos.

3.7.2.1. Usuarios

El módulo de usuarios se encuentra en el directorio “login/raw/clases/User.php”. Este archivo se encarga de la gestión de la cookie y la apertura del usuario así como de los permisos que tiene este mismo en toda la aplicación.

3.7.2.2. Nodos y enlaces

En el directorio “php/api-rest” encontramos los archivos cLink.php y cNode.php. Estos archivos contienen las clases Link y Node, encargadas de actuar como modelo respectivo de la información referente a sus correspondientes elementos almacenados en la base de datos.

Estas clases tienen unos atributos y un comportamiento similar y pretenden servir de patrón para futuras implementaciones de clases que respondan a necesidades de objetos almacenados en un futuro.

3.7.2.3. Otros Datos

El archivo “db.php” del directorio ‘/php’ contiene la clase DB que actúa de conector con la base de datos permitiendo realizar con su inclusión cualquier consulta mediante lenguaje PHP.

La clase logger, que se encuentra en el archivo “logger.php” del directorio “/php” es la encargada de registrar los errores de código PHP que puedan producirse a lo largo del uso de la aplicación en lado servidor.

La clase PHPMailer se encarga de proporcionarnos un servidor saliente de envío de correos para usos muy concretos como el aviso de instalación satisfactoria.

3.7.3. El Controlador

El modelo de la aplicación se compone de dos partes muy diferenciadas:

- Navegación.
- Obtención y manipulación de datos.

3.7.3.1. La interfaz de navegación

El archivo “index.php” en el directorio raíz es el encargado de servir cada una de las peticiones de páginas y direccionar al usuario a la correspondiente. Éste se encarga del procesado de todas las consultas que realiza el usuario teniendo en cuenta sus privilegios y su sesión.

Mediante variables GET y POST y de sesión en PHP realizaremos todas las posibilidades de navegación de la plataforma. (Ver figura 9)

3.7.3.2. Obtención y manipulación de datos (JSON API RPC)

Al tratarse de una aplicación de visualización de datos hay que tener siempre presente que además de las posibilidades de navegabilidad deberá existir un módulo encargado de la comunicación con los datos, que realice las funciones de obtención y manipulación de los mismos.

Las principales soluciones existentes son:

- Conexión directa mediante un driver PHP que enlace con la base de datos
    - Driver MySQL: API histórica.

    - Driver MySQLi: API nueva versión de su antecesora. Orientada a Objetos.
    - PDO: capa de abstracción de base de datos, con soporte para MySQL y otros sistemas gestores de bases de datos. Proporciona comandos preparados y otra serie de operaciones adaptadas al driver necesario
- Creación de una API
    - RPC (Remote Procedure Call): Protocolo de red que permite a la computadora ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambas. Supone un gran avance frente a los sockets utilizados hasta su invención. De esta forma el desarrollador se despreocupa de las comunicaciones. RPC pone énfasis en la diversidad de operaciones del protocolo o en los verbos.
    - REST (Representational State Transfer): Arquitectura software para sistemas hypermedia distribuidos como el World Wide Web. Termino originado en el año 2000 en la Tesis doctoral de Roy Fielding, unos de los principales autores del estándar HTTP y frecuentemente denominado RESTful. Este sistema enfatiza los recursos o substantivos, especialmente en los nombres que se le asigna a cada tipo de recurso.

Teniendo en cuenta los estándares web e influenciado por las nuevas corrientes en cuanto al manejo y manipulación de gran cantidad de datos en plataformas web la solución óptima para la herramienta, no cabe ninguna duda que sería la utilización de una API-RESTful. Cómo la mayoría del código de visualización se realiza mediante librerías Javascript como las librerías D3.js o la librería Javascript de Google Maps y los lenguajes utilizados tanto en lado cliente (Javascript) como en el lado servidor (PHP) tienen soporte para JSON, encontramos este sistema como el óptimo para la manipulación de información entre ambos lenguajes.

Las limitaciones temporales no nos han permitido desarrollar una API-RESTful. Teniendo en cuenta que el objetivo principal del proyecto no es precisamente el mecanismo de manipulación de los datos sino su visualización decidimos crear una API-RPC que atendiera a peticiones HTTP mediante variables GET y POST.

La aplicación dispone de las clases Users, Nodes, Links y RRDB encargadas de la obtención y manipulación de sus datos respectivos en la aplicación. Mediante el archivo ‘Ajax.php’ atenderemos estas peticiones ofreciendo múltiples respuestas para distintas consultas.

Las respuestas que ofrece este archivo tienen la siguiente estructura:

- Action (string): la acción que han atendido.
- Status (book): si la ha podido atender correctamente o no.
- Data (Object): datos obtenidos por esta instrucción.
- Url (String): Página de redirección en caso que se requiera.

Existen un número limitado de variables creadas para cada procedimiento. Las principal clasificación entre estas peticiones es según la atención de demandas de tipo consulta o de tipo modificación y manipulación de los datos. Esto se traduce a la variable DO y GET, imprescindible en cualquier consulta de este archivo y diferenciada por los permisos necesarios para cada tipo. Las peticiones de consulta las puede realizar cualquier usuario registrado mientras que las de manipulación y modificación tan solo las podrán realizar los usuarios administradores.

Responde a cada una de las peticiones de consulta mediante la instrucción GET. Algunos ejemplos son:

- "nodelist": devuelve la lista completa de nodos de la tabla índice.
- "nodetree": devuelve la jerarquía completa de los nodos o, en caso de consultar un nodo concreto, devuelve la jerarquía descendiente completa desde el nodo de la consulta.
- "node": devuelve la información del nodo consultado mediante su id.
- "link": devuelve la información del enlace consultado por su id.

"RRDB": devuelve la información de la tabla RRDB especificada mediante su id y su 'name'.

Responde a cada una de las peticiones de manipulación, alteración e inclusión de información mediante la instrucción DO. Por ese motivo, inicialmente realiza una comprobación de los permisos del usuario de la consulta mediante variables de sesión. Algunos ejemplos son:

- "node-add": Añade un nodo a la base de datos, la tabla índice.
- "node-edit": Comprueba si existe el nodo y en caso afirmativo modifica la información de éste.
- "node-delete": Elimina el nodo cuya id corresponda con la indicada.
- "hierarcyTree": Modifica los índices de la tabla nodos para crear la estructura jerárquica proporcionada.
- "createRRD": crea las tablas RRDB del nodo especificado.

3.7.4. La vista

En caso de no tener instalada la aplicación, la vista se encarga de presentar el formulario de instalación de la aplicación detallado en el apartado instalación de la aplicación de los anexos.

Una vez instalada, la vista principal en primer lugar se compone de dos secciones: sección interna y sección externa. Si no hemos iniciado sesión lo único que nos permitirá la sección externa será rellenar el formulario de acceso o nos anima a registrar nuestro usuario en la plataforma.

CAPITULO 4. DESARROLLO DE LA APLICACIÓN

4.1. Contexto

Realizar una aplicación no es en ningún caso una tarea trivial. Todos conocemos casos puntuales como el de Linus Torvalds, creador de Linux, Steve Jobs y Wozniak, fundadores de Apple, Bill Gates (Windows), Sergey Brin y Larry Page (Google), Mark Zuckerberg (Facebook) y otros más desconocidos como tal vez Jezz Brezos, fundador de Amazon. Estos casos, a pesar de todo, son aislados. Con un profundo respeto y admiración y salvando las distancias, la mayoría de tecnologías se desarrollan en equipo. Por ese motivo, tampoco habrá que obsesionarse con la idea de realizar una aplicación completa con un tiempo y personal tan limitado.

El proceso de desarrollo de software cuenta con 3 etapas:
1. Planificación.
2. Implementación, pruebas y documentación.
3. Despliegue y mantenimiento.

En nuestro caso, la última etapa no tendrá lugar debido a que este documento cerrará la segunda etapa.

Como modelo de desarrollo, teniendo en cuenta que tan solo había un desarrollador que concentraba todas las fases del desarrollo, podríamos decir que hemos elegido un modelo ágil. Este modelo utiliza un desarrollo iterativo o incremental que aboga por la construcción de secciones reducidas de software que irán ganando en tamaño para facilitar así la detección de problemas de importancia antes de que sea demasiado tarde.

Teniendo en cuenta estas premisas, realizaremos una repartición del tiempo en fases y seguiremos el modelo ágil, pudiendo revisar en cada momento cualquier decisión anterior. Las fases destacadas son:
- Especificación de requisitos.
- Diseño del software.
- Construcción e implementación del software
- Integración
- Validación.
- Despliegue.
- Mantenimiento.

Todas estas etapas, en el modelo de desarrollo incremental elegido, avanzarán en paralelo en la medida de lo posible.

4.2. El entorno de desarrollo

Dentro de la infinidad de ofertas de alojamiento que existen en internet, para el desarrollo del proyecto, decidimos alojarlo en un servidor de la empresa CDmon[40]. Esta elección fue sencillamente porque ya la hemos utilizado en anterioridad para múltiples trabajos -trabaja en Catalunya- por lo que el servicio de atención opera en el mismo horario y la asistencia es de calidad y no tiene un coste demasiado elevado en caso de elegir el servicio de alojamiento Linux compartido. Además, en esta oferta ofrecen bases de datos ilimitadas, lo que es muy interesante.

4.2.1. El logger

Como estamos desarrollando una aplicación en fase de producción, es muy conveniente guardar un registro de todos los posibles errores que pudieran suceder. PhpLogger será la herramienta encargada de guardar un registro diario de todos los errores en el lado servidor volcándolos en archivos de texto.

Realizaremos revisiones periódicas a estos archivos para estudiar el comportamiento de la aplicación. Cabe decir que a menudo encontraremos errores que sabremos identificar pero también puede ser que encontremos otro tipo de errores más complicados de localizar, que pudieran acarrear futuros problemas.

4.3. El workflow

4.3.1. Introducción

Durante la conferencia Webshaped de 2013, en Helsinki, Stephen Hay habló sobre la metodología de trabajo de los diseños "*responsive*"[30]. Este apartado no va estrictamente sobre los aspectos de la presentación a pesar de ir estrechamente relacionado con las fórmulas que introducía.

Hace escasamente cinco o seis años los proyectos web eran creados siguiendo las diferentes fases tal como indica la figura 10. En ella se refleja el clásico diseño en cascada. Aquel contexto no admitía demasiadas revisiones y lo que los clientes observaban no era más que mockups o diseños en Photoshop finalizados.

Figura 10 – Procesos del diseño web en cascada [41].

Este modelo tradicional había funcionado hasta entonces pero tenía un problema. El modelo de cascada no tiene ningún sentido al combinarlo con el diseño responsive. En realidad, este modelo nunca fue óptimo para el diseño web pero como todo el mundo estaba acostumbrado a utilizarlo y conocía sus distintas etapas y formas de entrega nunca nadie trató de perfeccionarlo.

4.3.2. La fórmula utilizada

El proceso más comúnmente utilizado en la actualidad al desarrollar diseños responsive es el que refleja la figura 11. Se basa en el artículo "*Responsive Summit: Workflow*" de Mark Boulton[31] y la presentación de Stephen Hay[32]. Este nuevo modelo refleja procesos más ágiles que el de cascada.

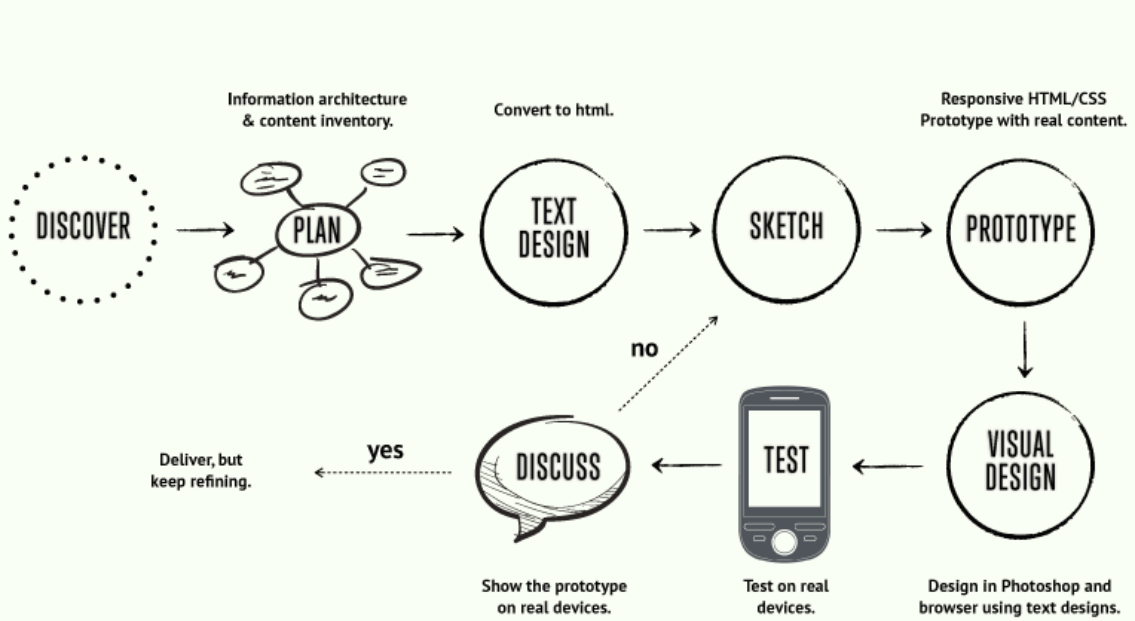


Figura 11 – Nuevo modelo de proceso de desarrollo [41].

4.3.2.1. Explicación del workflow

El proceso se inicia en la elaboración del contenido e incita a dedicar tiempo para realmente concentrarse en éste. Después se realizan unos pequeños borradores de contenido que se acostumbran a convertir en prototipos HTML, para abrirlos en diferentes navegadores móviles y observar cómo la solución responde a múltiples variables (dimensión de pantalla, aplicación, tecnología, …). También se elaboran sketches y diseños visuales antes y después del prototipo. Una vez elaborados los primeros sketches se retoman rápidamente los prototipos HTML y se empieza añadiendo algún CSS para comprobar si la idea está funcionando. El proceso entero se realiza en iteraciones que, a menudo, siguen el orden: sketch, prototipo, diseño, testeo, revisión. Este proceso entra en bucle hasta que el diseño funciona correctamente. Esta

ordenación se ha incluido de una forma lineal para simplificar el proceso, aunque no tiene por qué ser así en realidad.

4.3.3. Etapas seguidas

4.3.3.1. Case study

La primera etapa consiste en la recopilación de información y conocer el sector, el vocabulario, etc. El objetivo es obtener la mejor comprensión del contexto y los objetivos. Sin esta etapa es realmente imposible conocer las necesidades y requerimientos del proyecto.

4.3.3.2. Planificación

Este proceso se realiza en base al conocimiento previamente obtenido. Durante esta etapa se trata de definir el concepto, la arquitectura de la información y el árbol de navegabilidad del usuario. También incluye la concepción y la descripción de los elementos del contenido sin un nivel excesivamente preciso. Con una importancia relativa se desarrolla alguna plantilla sin demasiado detalle.

El apartado 1.3 de la memoria contiene el trabajo desarrollado durante este apartado comparando las herramientas existentes y sus características.

4.3.3.3. Diseño del contenido

El diseño del contenido significa acotar, reflejar y describir todos los conceptos del proyecto. Generalmente se elabora de forma escrita aunque puede reforzarse mediante alguna nube de ideas u otro esquema visual. Se considera una de las etapas más importantes de todo el proceso. El contenido es la razón por la que los usuarios utilizarán la aplicación de modo que todo el proceso dependerá de éste.

Este apartado incluye la definición, jerarquía y relación de los datos contenidos en el apartado 1.2 de esta memoria.

Acordamos que la aplicación consta de dos secciones principales: la de edición y la de visualización de los datos. Existirá una clara diferenciación de permisos para poder acceder a cada sección, siendo la primera de éstas únicamente accesible por los usuarios de perfil administrador. Para la navegación, entonces, hará falta almacenar variables de sesión y restringir las secciones con estas variables. La variables de sesión, entonces, no sólo servirán para restringir el acceso sino también para acotar funcionalidades una vez los usuarios hayan accedido a las sección interna.

En cuanto a la navegación, hay que tener en cuenta distintas variables. Una es la selección de sección (visualización/edición). Otra muy importante es la escala de tiempo a visualizar, que discutiremos más adelante. Por último, también se necesita disponer de una opción de abandonar sesión.

4.3.3.4. Elaboración de borradores (sketches)

El borrador es el proceso previo al salto a la plataforma de desarrollo. Dedicar un tiempo breve a la elaboración de un borrador nos puede ahorrar muchas horas de desarrollo posterior; no obtendremos un diseño definitivo pero nos puede ayudar a descartar ideas no funcionales.

El anexo V contiene algunos de los borradores de las etapas iniciales de la aplicación.

4.3.3.5. Elaboración del prototipo

El prototipo es la forma más realista de asegurar la reproducción de la idea sobre las diferentes plataformas del proyecto. También permite agilizar el feedback de la etapa de revisión.

4.3.3.6. Desarrollo completo pendiente de revisión

Este paso se hace mediante iteraciones sobre el prototipo. Introducimos el desarrollo de la lógica de la programación y reproducimos las recreaciones exactas de personalización del diseño visual.

4.3.3.7. Testeo

Es necesario incluir el testeo dentro del proceso para prevenir futuros problemas graves imprevistos. Este apartado incluye la documentación de distintas pruebas sobre múltiples plataformas, dispositivos y resoluciones de cada uno de los pasos anteriores.

4.3.3.8. Revisiones

En esta etapa presentaremos el proyecto y atenderemos las demandas y sugerencias que se precisen. Si no existieran ninguna de éstas consideraríamos el proyecto finalizado.

4.3.3.9. Iteración

Este proceso se somete iterativamente a las etapas descritas en anterioridad hasta conseguir la satisfacción de las demandas y requisitos.

En el capítulo 5 de la memoria recogemos las diferentes problemáticas encontradas durante el desarrollo del proyecto que han supuesto la constante iteración de este proceso.

4.3.3.10. Conclusiones sobre el proceso utilizado

No existe proceso de desarrollo de un proyecto perfecto. El aquí descrito tal vez no funcione para otro pero signifique un punto de partida de futuros métodos. Estas líneas pretenden animar al lector a ser crítico y constructivo con los procesos de trabajo para garantizar su evolución.

CAPITULO 5. RESULTADOS, LIMITANTES Y CONTINGENCIAS

5.1. El instalador

La aplicación requiere un instalador para su puesta a punto. Su desarrollo no supuso ningún imprevisto. El anexo VI detalla los aspectos fundamentales de su implementación.

5.2. La obtención de los datos

Para la inclusión de datos en este trabajo se intentó incluir información lo más veraz posible. A pesar de todo, este proceso ha sido una ardua tarea tanto por la dispersión de la fuente como la seguridad con que se encuentra. La inclusión de la información también es un proceso que, por mucho que deseemos facilitar mediante las más avanzadas técnicas y soluciones de diseño web, es muy lento debido a la gran cantidad de información y su nivel de detalle. La introducción de algunas redes ha supuesto muchas horas de trabajo.

En el anexo IX se incluyen algunas de estas referencias.

5.3. La sección interna

La sección interna consta de dos modos de operación. La edición y la visualización de datos.

5.3.1. Modo edición

Para este modo de acción desarrollamos una plantilla común para la gestión de nodos, enlaces y otro tipo de datos. Esta plantilla dispone de una tabla central que lista los elementos requeridos en filas y muestra sus propiedades en columnas. También tiene un filtro para la búsqueda específica de uno o más elementos. Existe un botón para la creación de nuevos elementos del mismo tipo.

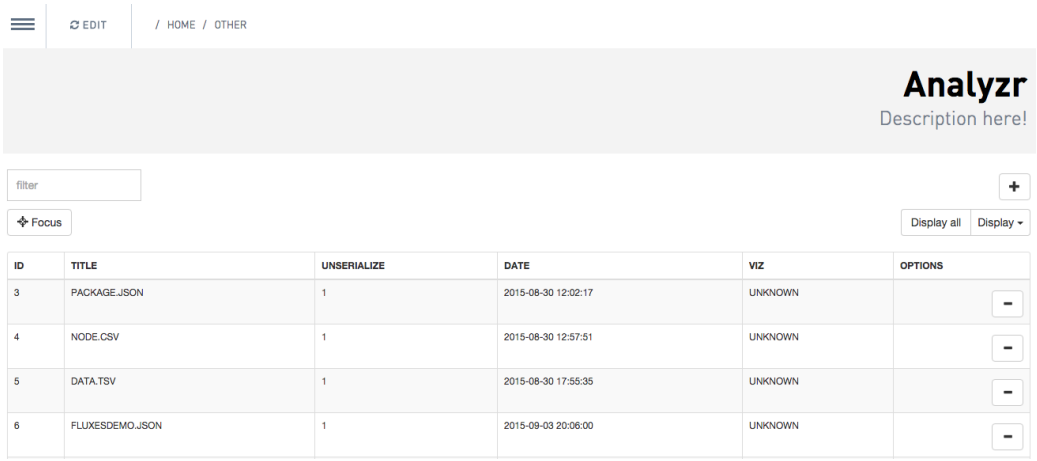


Figura 12 – Template de gestión y edición de elementos del proyecto.

Durante el proceso de testeo del template se presentó un incidente que no habíamos tenido en cuenta hasta entonces. Observamos que a partir de la inclusión de más de 200 elementos, el tiempo de carga de éste era superior a 10 segundos. Esto se debía a la gran cantidad de elementos a procesar en el DOM. Este problema lo resolvimos mediante la carga dinámica de los modal box de edición y eliminación de los elementos, reduciendo el tiempo a 1 segundo. Se propone otra solución a aplicar que consiste en la elaboración de un paginador de elementos que limite la carga a un número máximo de aproximadamente 30 elementos.

5.3.1.1. El creador y simulador de datos RRD

Debido a la ausencia de un proveedor continuo de datos, hubo que programar toda la lógica que conlleva desde cero. La activación de la creación de las tablas de datos así como su simulación se resumen en dos cómodos botones en el visualizador de nodos.

5.3.1.2. La gestión de la jerarquía

El caso particular de la gestión de los nodos (elemento central de agrupación de datos del proyecto) incluye una gestión jerárquica. Esta contingencia no había sido prevista inicialmente. Al realizar una investigación severa de las diferentes soluciones web de gestión de elementos jerárquicos encontramos que no existía ninguna posibilidad de reutilización de un código que la facilitara. Por este motivo, hubo que desarrollar una herramienta desde cero.

El editor muestra inicialmente un árbol jerárquico horizontal de los elementos introducidos en la base de datos. Permite recolocar los elementos arrastrando éstos por la pantalla y sentándolos sobre su superior. La figura 13 muestra este editor para el caso particular de la red AARNet.

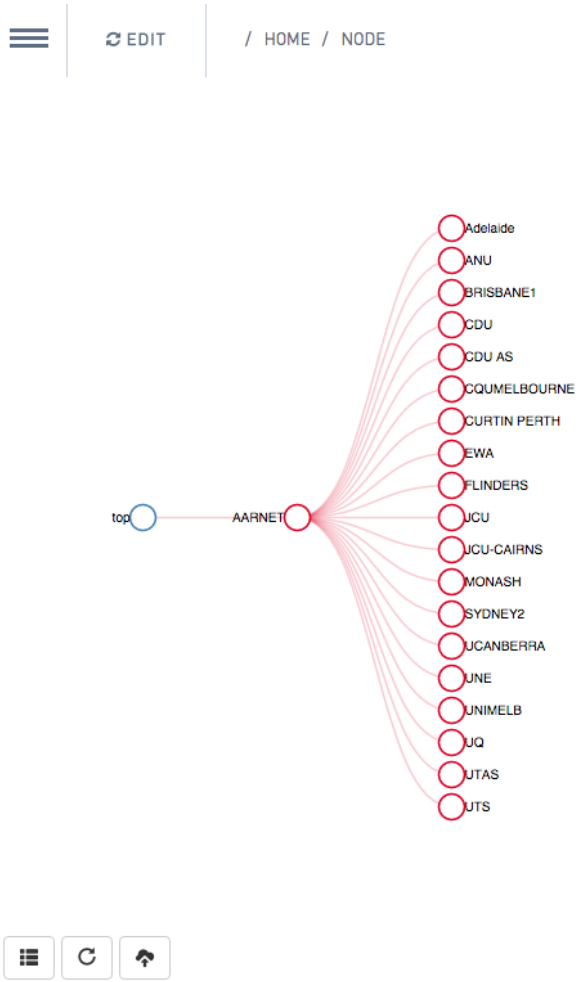


Figura 13 – Visualización de la edición jerárquica de la red AARNET.

En su parte inferior se incluyen tres botones que permiten volver a la lista de nodos, revertir los cambios y guardarlos en la base de datos respectivamente.

5.3.1.3. El importador (uploader)

Esta herramienta es una de las que puede suponer mayor vulnerabilidad a la estructura de la información. La inclusión de datos mediante archivos puede suponer un riesgo. Por este motivo confiamos en la adaptación de una tecnología ya existente para su implementación. Arfaly[22] realiza las comprobaciones pertinentes en el servidor impidiendo la manipulación del código mediante exploits en el lado cliente. También permite filtrar aquellos formatos de archivo no soportados (tan solo habilitamos los formatos JSON, CSV, TSV y SQL). Como Arfaly es un importador de archivos FTP al servidor, hubo que realizar unos hooks en su código que permitieran el parseo del formato y la validación del contenido antes de su inclusión serializada a la base de datos.

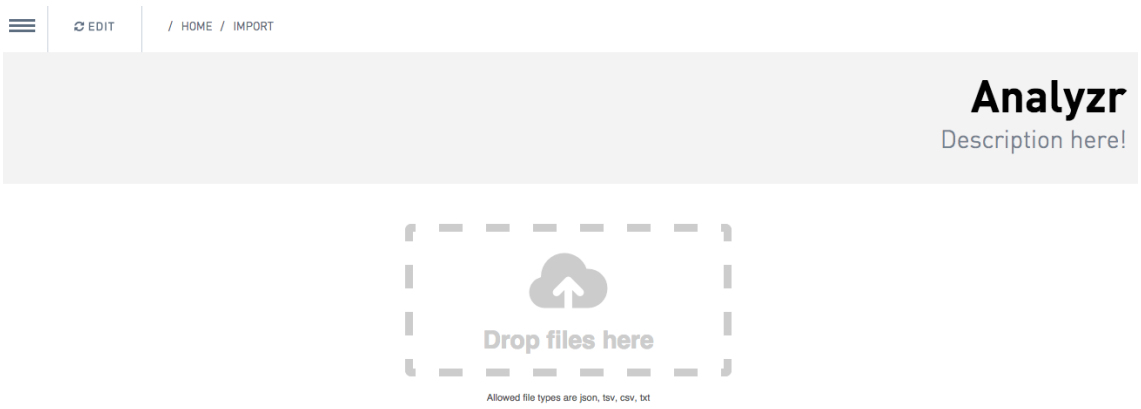


Figura 14 – Solución propuesta de impostación de datos.

Las limitaciones de tiempo nos han impedido desarrollar el clasificador de los datos importados en el proyecto, aunque sí incluye un verificador del contenido de los formatos SQL, JSON, CSV y TSV[33].

5.3.2. Modo visualización

Este es el objetivo principal del proyecto y por lo tanto es el que ha supuesto la mayor dedicación en su creación. Sigue la regla fundamental en los visualizadores: "*Overview first", zoom and filter, then details-on-demmand*"[7] descrita anteriormente

.

5.3.2.1. Visualización topológica

Realizamos una propuesta de lo que significaría la integración de la variable jerárquica y relacional a su vez sobre un modelo force-directed. En una primera instancia y sobre un número moderado de nodos (30 unidades) nos pareció un resultado satisfactorio. Al aumentar el número de nodos al escenario el resultado fue decepcionante pues no se conseguía ni comprensión de los datos ni estabilidad del visualizador ni precisión de los datos. A pesar de esto, intentamos mejorar los resultados incorporando un zoom al visualizador utilizando el scroll del mouse; supuso una ligera mejora. Al final consideramos que la mejor opción era prescindir de este modelo. (ver figura 15)

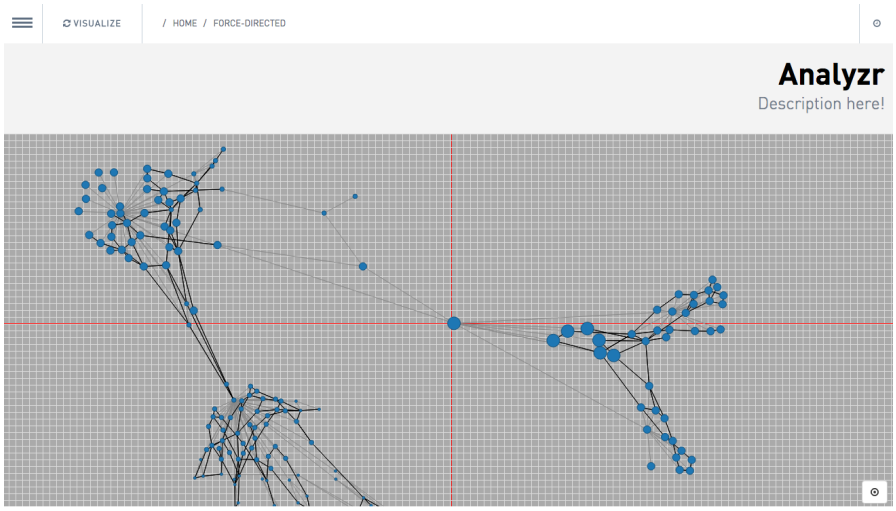


Figura 15 – Force-directed node map.

5.3.2.2. Visualización topográfica

Este visualizador es el elemento central de la aplicación. Para su desarrollo hemos combinado dos librerías importantes: API Google Maps en Javascript y D3.js.

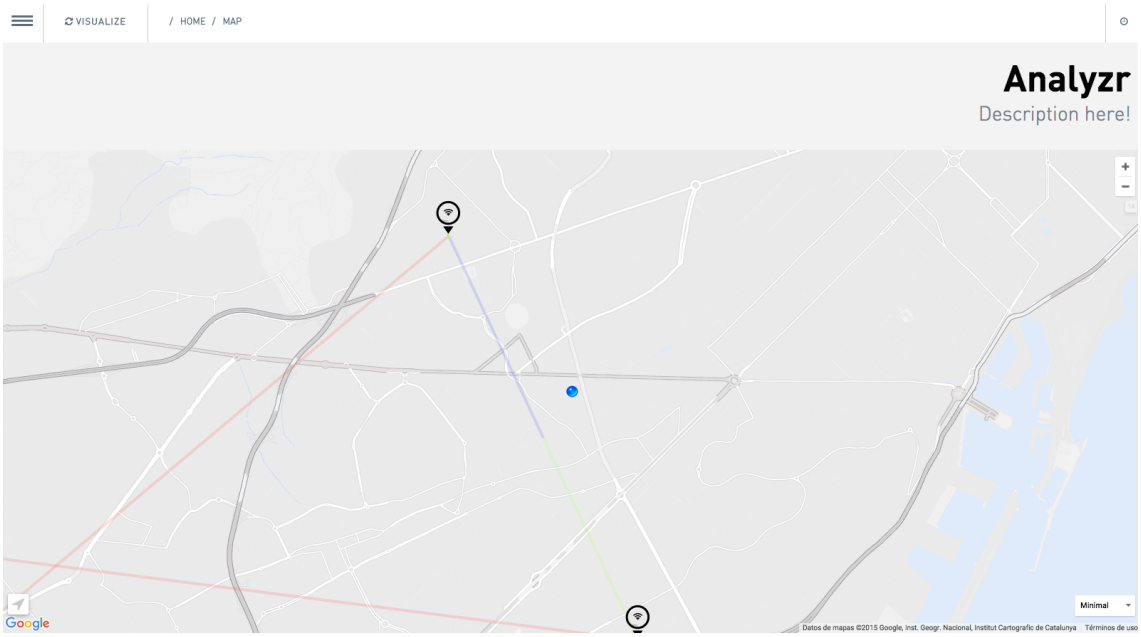


Figura 16 – Visualizador topográfico de los datos.

La API Google Maps dispone de 5 capas para la inclusión de elementos. Los tiles del mapa se cargan en la capa 2. Para conseguir la respuesta a eventos de click sobre los nodos tuvimos que insertar la información svg de d3.js en una capa más superficial del DOM, la 3. Aun así, existe una limitación en este modelo debido a la mala integración de ambas librerías. Los elementos cargados por D3js (nodos representados como círculos y enlaces como líneas) (ver figura 16) deben ser introducidos como elementos svg individuales para la correcta interpretación de las coordenadas geográficas por Google Maps. Esto

no supone un obstáculo para los nodos. Los enlaces, en cambio, suponen un problema en caso de desear programar su interactividad ya que el navegador sitúa el último elemento svg en la capa más superficial impidiendo interactuar con los anteriores. Resolvimos esta limitación concentrando toda la información en los nodos y habilitando un selector de carga dinámica de datos. (Ver figura 17).

El desarrollo de la aplicación tenía como objetivo representar las series temporales completas en el mapa. Durante el desarrollo encontramos que no había forma de procesar toda la información al mismo tiempo por parte del cliente. Por eso concentramos los esfuerzos en presentar esta información dinámicamente bajo demanda.

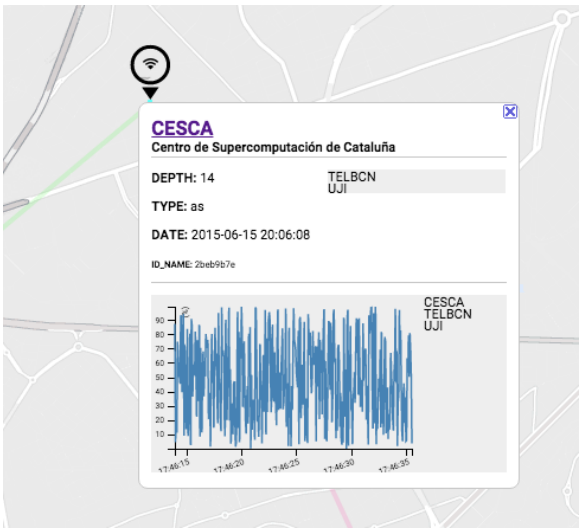


Figura 17 – Visualización detallada de la información del nodo CESCA de la red RedIRIS sobre el modelo topográfico.

5.3.2.3. Visualización mrtg del estado de los nodos y la carga de los enlaces

El objetivo de este visualizador era ofrecer la información detallada de los elementos nodo y enlace. Estos elementos comparten template pero muestran información diferente.

En una primera sección, se presenta la información del elemento en cuestión, ligeramente diferente debido a sus distintas características. (Ver figura 18 y 19)

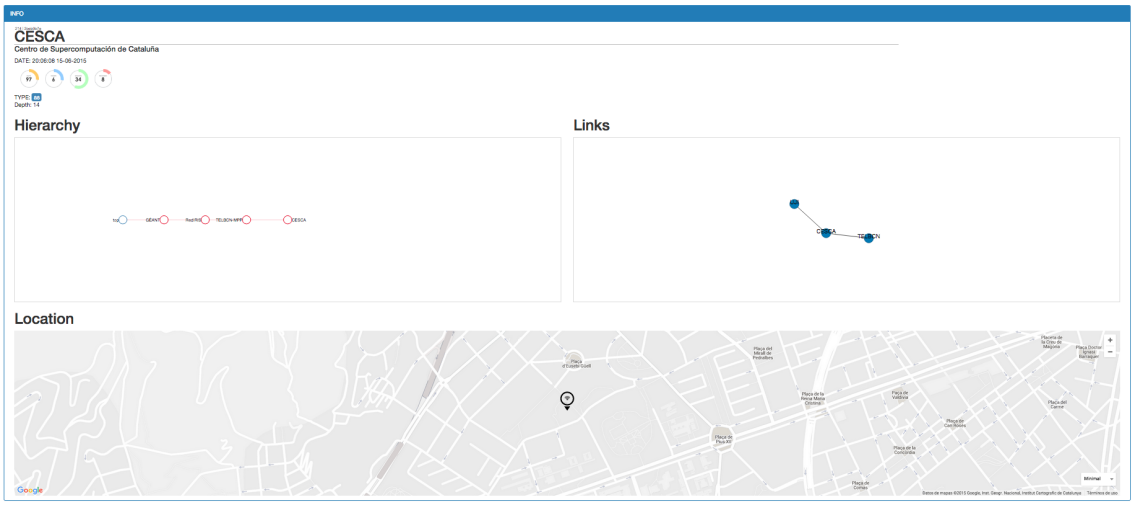


Figura 18 – Información detallada del nodo CESCA.

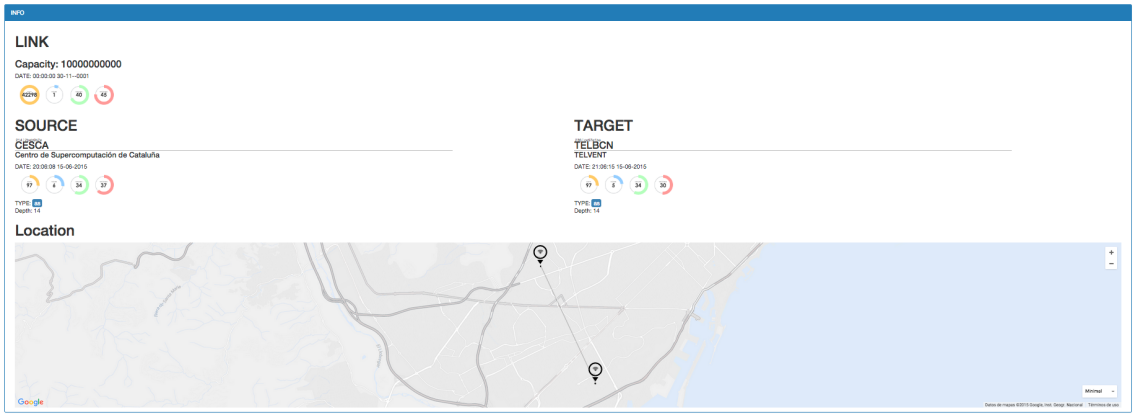


Figura 19 – Información detallada del enlace CESCA-TELBCN.

En segundo lugar presenta la visualización de la serie temporal a estudio para cada caso si dispone de información en cada escala de tiempo. Para el caso de los nodos, esta información es su rendimiento de CPU y está en valor relativo (%). En los enlaces, por el contario, se muestra su carga en valor absoluto (bps). Ver figura 20.

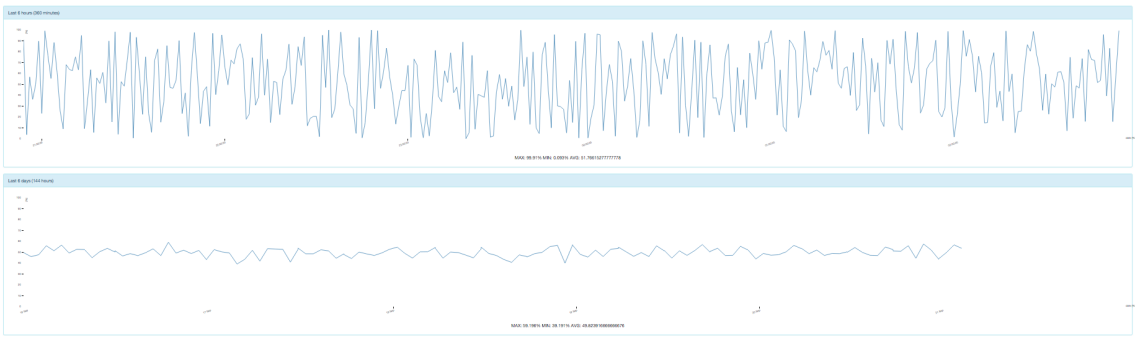


Figura 20 – Line-chart representativo de la carga del enlace en su evolución temporal para las escalas de tiempo minutera y horaria.

Este modelo es un competidor directo del MRTG y ofrece tres mejoras significativas:

- La carga dinámica de nuevos datos: la actualización se realiza por una consulta AJAX y se incluye en la serie temporal mediante una transición horizontal y ofrece una sensación de continuidad.
- Aporta nueva información jerárquica, relacional y topográfica al mismo tiempo.
- Permite la navegación entre elementos relacionados sin pasar por la visualización de nivel superior (la topográfica).

Un aspecto a considerar es la falta de sincronización entre el cliente y la introducción del nuevo dato ya que llegan mediante un temporizador minutero en el cliente. Se incluye, a pie de página, un botón capaz de forzar la obtención de datos y reiniciar el temporizador en caso de desear sincronizarlo con el disparador.

5.3.2.4. Visualización de datos de tráfico

Este visualizador recoge las ventajas de los modelos mat-chart y chord-diagram y supone un punto de partida para futuras implementaciones de visualización de flujos de datos. (Ver figura 21)

La sección derecha de la imagen corresponde con las visualizaciones de las entradas en la matriz de tráfico. La colorimetría es relativa al valor máximo encontrado en la totalidad del tiempo de estudio. Los valores de la matriz están representados en valor absoluto normalizando así la tabla de colores. Existe un acotador de tiempo de ventana en la parte superior para filtrar los tiempos de inicio y fin y poder estudiar la evolución temporal al detalle. La sección izquierda corresponde con la visualización de la carga relativa que supone en los enlaces cada uno de los saltos que intervienen en la transmisión de los valores de la matriz de entrada.

Consideramos interesante esta vinculación para la mayor comprensión de los procesos que intervienen en la congestión del tráfico en la red.

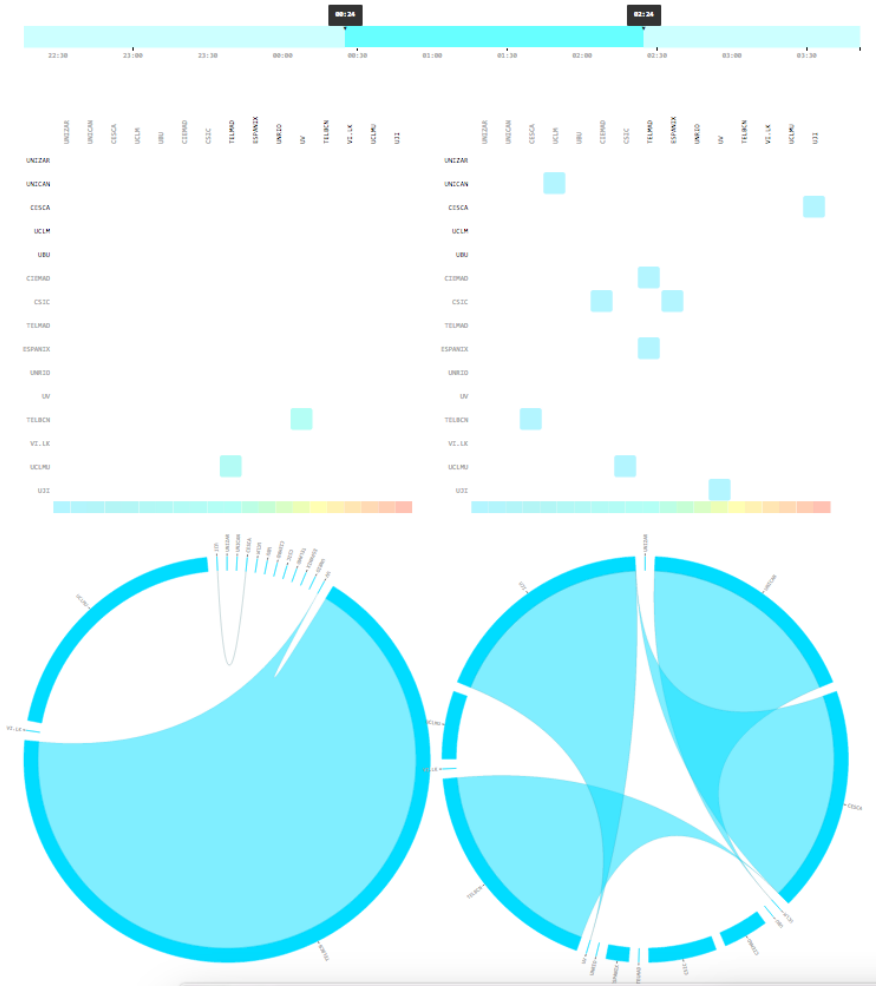


Figura 21 – Propuesta de visualizador de datos de la matriz de tráfico.

5.4. El generador de datos

Una de las mayores complicaciones a la hora de preparar el escenario fue el acceso de un flujo de datos. Como no encontramos ninguna fuente continua hubo que crear una desde cero. El resultado es muy satisfactorio ya que resuelve un problema complejo en dos cómodas acciones a nivel de usuario administrador. Este apartado se detalla en el anexo VIII.

5.5. La jerarquía de información.

Durante el desarrollo del proyecto encontramos un detalle que no habíamos contemplado inicialmente. Los nodos tienen una estructura jerárquica por lo que la base de datos debía soportar este requerimiento. El anexo VII incluye los detalles de la evolución del modelo.

CAPÍTULO 6. CONCLUSIONES

El objetivo fundamental de este trabajo era desarrollar una herramienta que proporcionara distintas soluciones para la visualización de datos de red. En la fase embrionaria del proyecto pretendíamos que le la fuente fuese NEMO[47] pero desvincularnos de él, por falta de tiempo, supuso un nuevo reto. El principal escollo que hemos encontrado ha sido utilizar una fuente de datos que se amoldara con las necesidades del proyecto.

Aunque hemos incluido una solución a esta problemática mediante un simulador hubiera sido preferible disponer de una fuente fidedigna. Nos hubiera facilitado la labor de desarrollo y también hubiera maximizado el tiempo dedicado al diseño y programación de más modelos de visualización.

Las nuevas corrientes de diseño web están mejorando las formas de interactuación de los usuarios. Esto posibilita el acceso al contenido de forma más agradable, demostrando que cualquier tarea puede realizarse de una manera cada vez más sencilla.

Los lenguajes y las librerías utilizadas en el proyecto han sido acertados. D3.js es una librería muy potente en web y ofrece un abanico nuevo de posibilidades de interactuación del usuario con el navegador. Hemos encontrado alguna complicación a la hora de interactuar con otras librerías. A pesar de todo fue posible encontrar una solución que consiguiera obtener los resultados deseados.

Es imprescindible considerar la carga dinámica de información para cualquier solución de este tipo de visualizadores debido al elevado número de datos.

La aplicación supone un punto de partida en la utilización de herramientas web modernas para visualizar datos de tráfico. Ha sido desarrollada utilizando tecnología moderna y pensando siempre en un uso, plural y versátil. Su integración es muy cómoda y bastaría con unos sencillos cambios en el código para adaptarla a otros sistemas.

También ha sido desarrollada modularmente, de manera que sea muy simple realizar ampliaciones.

La principal vía de desarrollo futura debería acometer el motor de comunicación con los datos. El uso de la API-RPC queda muy limitado y el desarrollo de una API-RESTful sería un objetivo importante considerar. Otro de los objetivos futuros sería la creación de nuevos modelos de visualización sobre la nueva API.

La utilización de herramientas de visualización que aporten facilidades a la experiencia del usuario y mejoren la comprensión permitirá minimizar los tiempos de respuesta a caídas y reconfiguraciones de las redes. También supondrá mejoras en la optimización de la configuración de la red. Todas estas

conclusiones podrían suponer una mejora en el consumo eléctrico global, optimizando el impacto medioambiental.

Personalmente, este proyecto me ha servido para mejorar mi conocimiento en los distintos lenguajes web. También me ha aportado una nueva metodología de trabajo mucho más eficaz. La metodología iterativa admite mucha más comunicación en las fases del trabajo donde todas avanzan en paralelo.

CAPÍTULO 7. BIBLIOGRAFÍA

[1] Hal Varian "*How the Web challenges managers*" McKinsey & Company. http://goo.gl/mdl7ux Jan 2009.

[2] Kirk, A. "*The 8 hats of data visualization design*". Visualising Data. http://goo.gl/DuM6W0 June 26 2012.

[3] Kirk, A. "*Data Visualization: A Sucessfull Design Process*". PACKT Publishing. December 2012.

[4] Few, S. "*Show me the Numbers: Designing Tables and Graphs to Enlighten*". Copyrighted Material. 2nd Ed. Jun 2012.

[5] Few, S. "*Telling Compelling Stories with Numbers*" SLDS Annual Grantee Meeting. (p71-73). https://goo.gl/YdHxv5 November 14, 2008.

[6] Huff, D. "*How to lie with statistics*". W.W. Norton & Company, Inc. Oct 1993.

[7] Card, S.K., Mackinlay J.D. and Shneiderman, B. "*Readings in Information Visualization: Using Visions to Think (Interactive Technologies)*". Morgan Kaufmann Publishers. Feb 1999. (p625).

[8] Fry, Benjamin. "*Computational Information Design*". Masachusets Institute of Technology. Apr. 2004

[9] Dubakov, M. "*Visual Encoding*". https://goo.gl/pLu6Ci September 2012.

[10] Bostock, M. "*D3 Showreel*". http://goo.gl/HhlYji

[11] "*Salesforce Training*". https://goo.gl/yvNdKg

[12] Bostock, M. "*Radial Reingold-Tilford Tree*" http://goo.gl/MQ2R3y. November 2012.

[13] Bostock, M. "*Zoomable Treemap*". http://goo.gl/mPOkQe

[14] Davies, J. "*Cofee flavour weel*" https://goo.gl/qt7QN3

[15] Graves ,A. "*D3js Radial chart example*" http://goo.gl/KJYAsq

[16] Circos software package for visualizing information web page. http://circos.ca/

[17] Krywinski, M. Hiveplots – Rational network visualizations – Farewell to hairballs. http://www.hiveplot.net/

[18] Krywinski, M. Birol, I. JM. Jones, S and A. Mara, M. "*Hiveplots – rational aproach to visualizing networks*". Biefings in Bioinformatics. Dec 2011.

[19] Lucas, A. "*Tools for visualization*" Interactive visualization graph. http://goo.gl/7bqJQH Sep, 2015.

[20] Sennhausser, O. "*Round Robin Database Storage Engine (RRD)".* GNU FDL. http://goo.gl/iJLkHe

[21] Planificación Round Robin. Wikipedia. https://goo.gl/4W4ybN Sep 2105.

[22] "*Arfaly.js – Powerfull & rsponsive multi file uploader*". Mindsquare. http://goo.gl/LjorCv

[23] "*Premium Login – Authentication File System*". EdcoreWeb. http://goo.gl/Cw6qJZ

[24] "*MySQL Conversion Class*". ToolkitSystems. http://goo.gl/oj4Ta5

[25] "Tympanus" Codrops. http://goo.gl/T8stiK

[26] Lou, M. "*Fullscreen Form Interface*" Tympanus Codrops. http://goo.gl/TH5eOu Jul. 2014.

[27] Giraudel, H "*Custom Login Form Styling*" Tympanus Codrops. http://goo.gl/v596du Oct. 2012.

[28] Lou, M. "*Google Nexus Website Menu*" Tympanus Codrops. http://goo.gl/uFwzzu Jul 2013.

[29] Lou, M. "*Fullscreen Overlay Effects*" Tympanus Codrops. http://goo.gl/9oazdO Feb 2014.

[30] Hay, S. "Responsive Worflows". Webshaped Conference. https://goo.gl/RSk1ST May 2013.

[31] Boulton, M. "Responsive Summit: Workflow" http://goo.gl/AOrihE Feb 2012.

[32] Hay, S. "Responsive design Workflow" Mobilism conference presentation. http://goo.gl/ghZkS6 Oct 2012.

[33] James Reese, R. "*TSV File to PHP Array*". https://goo.gl/5P1ujD Jun 2015.

[34] Celko, J. "*Joe Celko's Trees and Hierarchies in SQL for smarties*" Morgan Kaufhann Publishers. Apr. 2004.

[35] DB-Engines Ranking http://goo.gl/gk2gbt Sep. 2015.

[36] DB-Engines Popularity Trend http://goo.gl/1yjgwh Sep. 2015.

[37] Jiménez, C. and Rincón, D. http://goo.gl/ICOM19 UPCOMMONS. Jun. 2012.

[39] Minguillón, I. And Rincón, D. http://goo.gl/RXqTIC UPCOMMONS. Gen. 2011.

[40] CDMON https://goo.gl/0LiSEL Jun 2015.

[41] Viljami, S. http://goo.gl/D3OJsO "*Responsive Workflow*". May 2012.

[42] Rediris National Network http://goo.gl/3TMvJ6 RedIRIS Jun 2015.

[43] GÉANT tools https://goo.gl/8Niuc3 GÉANT Jul 2015.

[44] AARNET National Intercapital Network https://goo.gl/dJZD7N Jun 2015.

[45] AARNET International Network Map https://goo.gl/05kPdM Jun 2015.

[46] Topology Zoo Network files http://goo.gl/YeeoRP Jul 2012.

[47] Raspall, F. and Rincón, D. "Bulding NEMO, A Tool to track routing in IP Networks". I2Cat Foundation / UPC-BarcelonaTech. http://goo.gl/ukrvtm (p31-33) Apr. 2013.

[48] Sistemas de Gestión de Bases de datos Relacionales https://goo.gl/I49B4E Wikipedia. Sep. 2015.

[49] D3js Data Driven Documents http://d3js.org/

GLOSARIO

AJAX: Asynchronous JavaScript And XML.

API: Application Programming Interface.

CSV: Comma-Separated Values

D3: Data Drive Documents

DOM: Document Object Model.

FTP: File Transfer Protocol.

GUI: Graphics User Interface.

IIS: Internet Information Services.

JSON: JavaScript Object Notation.

MRTG: Multi-Router Traffic Grapher.

NREN: National Research and Education Network.

RRD: Round Robin Database.

SNMP: Simple Network Management Protocol.

SQL: Structures Query Language.

TSV: Tab-Separated Values.

UI: User Interface.

URL: Uniform Resource Locator.

ANEXO I: SISTEMAS ADMINISTRADORES DE BASES DE DATOS RELACIONALES

Cualquier sistemas de gestión de bases de datos relacionales debe atender 12 reglas para ser considerado de este tipo [48]:

Regla 0. Debe ser relacional. Una base de datos y un sistema de gestión. Debe utilizar sus capacidades relacionales exclusivamente para gestionar una base de datos.

Regla 1. Regla de la información. Toda su información debe estar representada explícitamente en el esquema lógico.

Regla 2. Regla del acceso garantizado. Todos los datos contenidos deben ser accesibles a nivel lógico utilizando una combinación de nombre de tabla, valor de clave primaria, y nombre de columna con garantía.

Regla 3. Regla de tratamiento sistemático de valores nulos. Para la representación de información desconocida o no aplicable de manera sistemática se deben soportan valores nulos.

Regla 4. Diccionario dinámico en línea basado en el modelo relacional. La descripción de la base de datos se representa a nivel lógico de la misma manera que los datos normales, de modo que los usuarios autorizados pueden aplicar el mismo lenguaje relacional a su consulta, igual que lo aplican a los datos normales.

Regla 5. Regla de sub-lenguaje de datos completo. debe existir al menos un lenguaje cuyas sentencias sean expresables, mediante una sintaxis bien definida, como cadenas de caracteres y que sea completo.

Regla 6. Regla de actualización de vistas. Todas las vistas que son teóricamente actualizables se deben actualizar por el sistema.

Regla 7. Inserción, actualización y borrado de alto nivel. La capacidad de manejar una relación base o derivada como un solo operando se aplica tanto a las consultas como a la inserción, actualización y eliminación de datos.

Regla 8. Independencia física de datos. Los programas de aplicación y actividades del terminal permanecen inalterados a nivel físico cuando quiera que se realicen cambios en las representaciones de almacenamiento o métodos de acceso.

Regla 9. Independencia lógica de datos. Los programas de aplicación y actividades del terminal permanecen inalterados a nivel lógico cuando quiera que se realicen cambios a las tablas base que preserven la información.

Regla 10. Independencia de integridad. Los limitantes de integridad específicos para una determinada base de datos relacional deben poder ser definidos en el sub-lenguaje de datos relacional, y almacenables en el catálogo, no en los programas de aplicación.

Regla 11. Independencia de distribución. Una base de datos relacional tiene independencia de distribución.

Regla 12. Regla de la no-subversión. Si un sistema relacional tiene un lenguaje de bajo nivel (un registro de cada vez), ese bajo nivel no puede ser usado para saltarse (subvertir) las reglas de integridad y los limitantes expresados en los lenguajes relacionales de más alto nivel (una relación (conjunto de registros) de cada vez).

Este conjunto de reglas ha abierto a día de hoy infinidad de alternativas a la hora de escoger entre distintos modelos de base de datos.

DB-Engines (Knowledge Base of Relational and NoSQL Database Management Systems) es una iniciativa que reúne la información actualizada relacionada con los sistemas de gestión de bases de datos. Se encarga de confeccionar un ranking mensual basado en la popularidad de estas tecnologías[35].

282 systems in ranking, September 2015

Rank			DBMS	Database Model	Score		
Sep 2015	Aug 2015	Sep 2014			Sep 2015	Aug 2015	Sep 2014
1.	1.	1.	Oracle	Relational DBMS	1463.37	+10.35	-3.53
2.	2.	2.	MySQL	Relational DBMS	1277.75	-14.28	-19.39
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1097.83	-10.83	-111.04
4.	4.	↑ 5.	MongoDB +	Document store	300.57	+5.91	+59.58
5.	5.	↓ 4.	PostgreSQL	Relational DBMS	286.18	+4.31	+30.38
6.	6.	6.	DB2	Relational DBMS	209.14	+7.91	+12.11
7.	7.	7.	Microsoft Access	Relational DBMS	146.00	+1.79	+5.52
8.	8.	↑ 9.	Cassandra +	Wide column store	127.60	+13.60	+39.74
9.	9.	↓ 8.	SQLite	Relational DBMS	107.66	+1.84	+15.04
10.	10.	↑ 12.	Redis +	Key-value store	100.65	+1.85	+26.05
11.	11.	↓ 10.	SAP Adaptive Server	Relational DBMS	86.52	+1.41	+1.10
12.	12.	↓ 11.	Solr	Search engine	81.94	+0.04	+6.17
13.	13.	13.	Teradata	Relational DBMS	74.27	+0.68	+8.11
14.	14.	↑ 16.	Elasticsearch	Search engine	71.55	+1.91	+30.03
15.	15.	15.	HBase	Wide column store	59.03	-0.92	+14.00
16.	16.	↑ 19.	Hive	Relational DBMS	53.53	-0.35	+22.12
17.	17.	↓ 14.	FileMaker	Relational DBMS	51.00	-0.87	-1.64

Figura 22 – Ranking mensual de DB-Engines[35]. Sep. 2015.

Las posibilidades que nos ofrecía el servidor del escenario eran MySQL y PostgreeSQL. Éstos tenían características similares para los requisitos de la aplicación. Cómo buscábamos un sistema de gestión popular y extendido y que garantizara todos los requerimientos, elegimos el primero que apareciera en la lista y que soportara nuestro escenario: MySQL. (Ver figura 22)

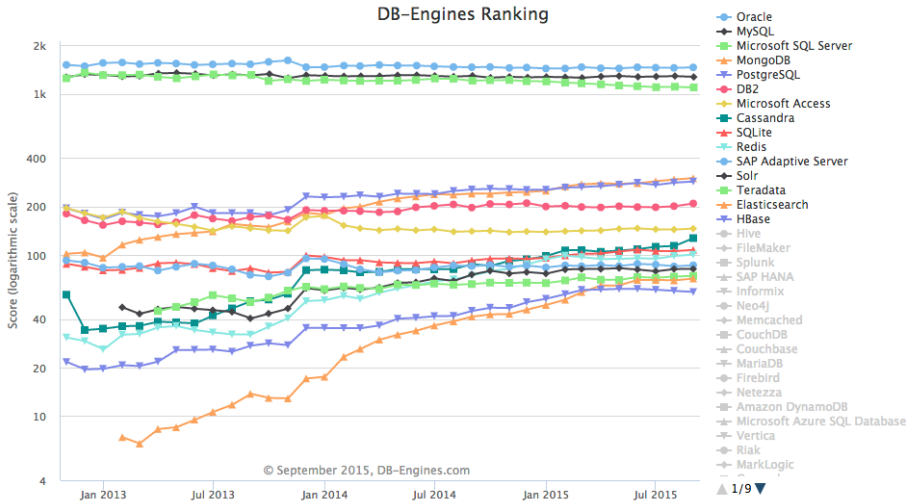


Figura 23 – Evolución temporal de la popularidad de los sistemas de gestión de base de datos[36].

Teníamos que asegurarnos que esta clasificación no reflejaba un valor singular si no que este sistema figuraba en esta posición durante bastante tiempo. DB-Engines también facilita un gráfico de la evolución temporal (ver fig. 23) donde pudimos constatar que nuestra elección era la más indicada.

ANEXO II: INTRODUCCIÓN AL D3

II.1. ¿Qué es?

Fundamentalmente, D3[49] es una elegante herramienta de software que facilita la generación y manipulación de documentos web con datos. Esto lo hace de forma que:

1. Carga los datos en la memoria del navegador.
2. Enlaza datos a elementos del documento, creando nuevos elementos necesarios.
3. Transforma estos elementos interpretando cada uno y proporcionándole las propiedades adecuadas.
4. Conmuta elementos entre estados según las acciones del usuario.

Aprender D3 es un proceso sencillo de asimilación de la sintaxis necesaria para ordenar a la máquina la carga los elementos, la transformación al plano visual y la transición correspondiente a cualquier evento del navegador.

El proceso de transformación es el más importante y es dónde sucede el mapeo. D3 proporciona la estructura para aplicar estas transformaciones pero hay que definir las reglas de mapeo. Todas las decisiones visuales las realiza el programador. Debemos proporcionar el concepto, elaborar las normas y D3 las ejecutará

Protovis ofrecía la creación de representaciones de la forma más sencilla posible incluso para aquellos usuarios sin experiencia previa en programación. A pesar de esto, el usuario tenía que crear una capa de representación abstracta. El diseñador podía abordar esta capa utilizando la sentencia 'protovis', pero no era accesible a través de métodos estándar, lo que hacía complicada su depuración. El aprendizaje de esta herramienta implicaba la especialización en un lenguaje específico y único.

Por este y otros motivos, en 2011, Mike Bostock, Vadim Ogievetsky y Jefff Herr introdujeron oficial y definitivamente D3. A diferencia de Protovis, D3 no requería una capa abstracta intermedia para funcionar sino que operaba directamente sobre el documento por sí misma. Esto significó un avance a la hora de depurar, además de una mejor experiencia de usuario y un sustancial sinfín de posibilidades visuales. La única contra es que requería de una curva de aprendizaje más pronunciada.

II.2. ¿Qué no es?

Cabe decir que esta librería está destinada con unos propósitos que no le permiten lo siguiente:

1. D3 no genera visualizaciones redefinidas por ti (los layouts D3 son una ligera excepción, ya que ayudan a conseguir visualizaciones recurrentes de forma rápida).
2. D3 no pretende ser soportado por antiguos navegadores. Esto ayuda a mantener el código limpio, libre de hacks que ofrezcan soporte a antiguas versiones de Internet Explorer, por ejemplo, o navegadores que no cumplan estándares. La filosofía es que creando herramientas y rechazando el soporte a antiguos navegadores se centra en animar a la gente a actualizarse (en lugar de impedir el proceso, que con ello nos exige seguir soportando estos navegadores y así sucesivamente, entrando en un círculo vicioso).
3. D3 no soporta map tiles de bitmap cómo los provistos por Google Maps o Cloudmade. D3 es perfecto para cualquier elemento vectorial – imágenes SVG o datos GeoJSON – pero jamás permitirá trabajar con los tradicionales map tiles. Cuando nos encontramos realizando Geo-mapping podemos utilizar ambas, SVG incrustado con D3 y otra herramienta con tiles, o probar de encontrar la forma de integrar D3 en la capa de datos y la otra herramienta para la base de map tiles.
4. D3 no oculta los datos de origen. D3 está programado en Javascript, código que se ejecuta en el lado cliente y por tanto, los datos a interpretar y visualizar deben ser enviados a este. Otras alternativas incluyen utilizar herramientas propietario (cómo Flash) o visualizaciones pre-renderizadas cómo imágenes estáticas (cómo PHP Network Weathermap) y enviar éstas al navegador. D3 plantea la pregunta siguiente: ¿Si no estamos interesados en enviar datos, entonces, porqué, por el contrario, sí nos puede interesar compartir la representación de estos? El propósito de la representación es comunicar los datos de forma clara y concisa y con objetividad así que elegir la transparencia es mejor solución que guardar temor al robo u obtención de datos.

II.3. Alternativas

D3 no es la librería definitiva y perfecta para todos los proyectos. Algunas veces tan solo necesitamos de un gráfico sencillo y no tenemos tiempo de programar código desde cero.

Para estas situaciones, puede resultar una buena idea conocer otras herramientas del panorama.

En el mundo del código de programación y particularmente en el de web, existen multitud de alternativas. Aquí presentamos algunas de éstas, todas ellas usuarias de estándares web y gratuitas, listas para su descarga y utilización.

II.3.1. Gráficas Sencillas

Para el desarrollo de gráficas sin grandes conocimientos de programación encontramos las herramientas:

Flot
Librería de visualización para jQuery que utiliza elementos canvas HTML y soporta navegadores antiguos inclusive Internet Explorer 6.

gRaphaël
Librería de gráficos de barras basada en Rapäel (mirar a continuación). Utiliza canvas y también soporta viejos navegadores.

Highcharts JS
Librería basada en Javascript con multitud de plantillas para representaciones gráficas. Utiliza SVG para navegadores modernos y VML para versiones antiguas.

jqPlot
un plugin para mostrar gráficos con jQuery. Suporta IE7 y recientes.

Plety
Un plugin jQuery para sencillos gráficos de barras, líneas y circulares. Tan solo soporta navegadores nuevos

Timeline.js
Una librería específicamente creada para generar líneas de tiempo.

II.3.2. Representaciones gráficas

Un gráfico no es más que la estructuración organizada de la relación entre datos.

Arbor.js
Una librería de gráficos que utiliza jQuery.

Sigma.js
Una librería ligera para la representación de gráficos.

Geomapping
Diferenciando entre mapping (todas las representaciones son relacionales) y geomapping (representaciones que incluyen datos geográficos, geodata o mapas tradicionales). D3 tiene un sinfín de funcionalidades de geomapping, pero debemos conocer algunas alternativas.

Katograph
Librería Javascript y Python para mappeos enormes y completa-

Leaflet
Una librería para mapas de mosaico diseñado para la interacción fluida

Modest Maps
El gran introductor de las librerías de mapas en mosaico. Ahora

mente vectorizados creada por Greor Aisch.

en navegadores y dispositivos móviles. Añade algún soporte para mostrar capas de datos en svg sobre los mapas en mosaico.

disponible en múltiples plataformas. En la actualidad ha sido sucedido por Polymaps.

Polymaps
Una librería para la creación de mapas en mosaico. Soporta capas SVG sobre sus mosaicos.

II.3.3. Otras alternativas para programar de cero

Estas herramientas, al igual que D3, nos proporcionan métodos de representación de formas visuales pero sin pre-diseño de plantillas. Si lo que deseamos es empezar desde cero cualquiera de estas soluciones puede ser ideal.

Processing.js
Una implementación nativa en Javascript de Processing , el maravilloso lenguaje de programación para artistas y diseñadores nuevos en el mundo de la programación.

Paper.js
Es un framework para la renderización de gráficos vectoriales en canvas. Además, su web es una de las más bellas de internet y sus demos son increíbles.

Raphaël
Otra librería para plasmar gráficos en canvas. Popular por su sintaxis amigable. Soporta antiguos navegadores.

II.3.4. Tri-dimensionales

D3 no es la mejor librería en 3D, sencillamente porque los navegadores web son históricamente en dos dimensiones. Debido al creciente soporte del WebGL, existen ahora muchas alternativas para generar experiencias 3D.

PhiloGL
Una herramienta WebGL específicamente creada para visualizaciones 3D.

Three.js
Una librería para la generación de cualquier tipo de escena 3D que podamos imaginar, producida por el equipo de artistas de datos de Google.

II.3.5. Herramientas construidas mediante D3

Cubism
Un plugin de D3 para la visualización de series temporales de datos, escrito por Mike Bostock.

Dashku
Una herramienta para crear cuadros de mando en tiempo real y widgets utilizando HTML, CSS y JavaScript.

NVD3
Plantillas reutilizables en D3

Rickshaw
Otra herramienta para la visualización de series temporales de datos

Tributary
Tributaria es un entorno experimental para la creación rápida de prototipos de código de visualización escrita por Ian Johnson. Proporciona varias bibliotecas útiles, como D3, así como una interfaz sencilla para la edición de código en vivo.

ANEXO III: LA COLORIMETRÍA EN EL PROYECTO

La utilización de distintas técnicas para la fácil comprensión de los resultados por parte de un usuario de éstos son generalmente las que utilizan el sentido más desarrollado por el ser humano: la vista. Por este motivo la repartición de las distintas variables a estudio se reparten utilizando elementos de interpretación visual, del que destacamos el espacio y el color.

El primer elemento permite la reproducción de distintos modelos de visualización permitiendo cómodamente la interpretación en 3 dimensiones o incluso más en algún caso más actual (hiveplot[17] – circos[16]).

Otro de las técnicas más comunes de utilización es el color. Esta herramienta es la más cómoda para la interpretación por parte del ser humano, utilizándola a diario y de forma instintiva. Por ese motivo es una técnica utilizada en todos los modelos de visualización. Definimos el color cómo la percepción visual que genera el cerebro humano y de otros animales al interpretar las señales nerviosas que envían los foto-receptores de la retina ocular interpretando y diferenciando las distintas longitudes de onda que captan de la parte visible del espectro electromagnético. Existe una necesidad de estandarizar el color y poderlo clasificar y dividir de manera estándar. El procedimiento utilizado en la medida del color consiste sustancialmente en sumar la respuesta de estímulos de colores y su normalización a la curva espectral de respuesta del foto-receptor sensible al color. La colorimetría es la ciencia que estudia la medida de los colores y desarrolla métodos para la cuantificación del color (obtención de valores numéricos del color).

III.1. La percepción humana del color

La percepción del color en el ojo humano se produce en las células sensibles de la retina denominadas conos. Hay tres tipos de conos: los que captan la luz roja, la verde y la azul, es decir, los tres colores primarios aditivos, con cuya combinación podemos percibir toda la gama de colores; por esto se considera que la visión humana es tri-cromática tal y como indica la figura 24.

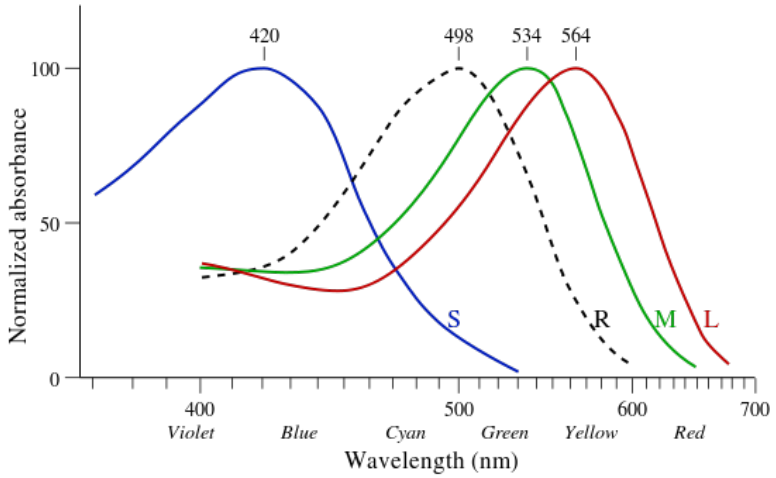


Figura 24 – Curvas de absorción de color en función de su longitud de onda para cada cono (S, M, L) y bastones (R)

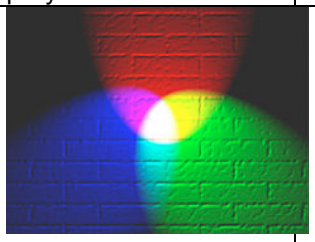
Las células sensoriales de la retina reaccionan de forma distinta a la luz y a su longitud de onda. Los bastones se activan en la oscuridad, y sólo permiten distinguir el negro, el blanco y los distintos grises. Los conos sólo se activan cuando los niveles de iluminación son suficientemente elevados. Los conos captan radiaciones electromagnéticas, rayos de luz, que más tarde darán lugar a impresiones ópticas. Los conos son acumuladores de cuantos de luz, que transforman esta información en impulsos eléctricos del órgano de la vista. Hay tres clases de conos, cada uno de ellos posee un fotopigmento opsina que sólo detecta unas longitudes de onda concretas, aproximadamente las longitudes de onda que transformadas en el cerebro se corresponden a los colores azul, rojo y verde. Los tres grupos de conos mezclados permiten formar el espectro completo de luz visible y son los siguientes:

- Cono L: captación de ondas largas (650 nm) correspondiente a la luz roja y mediante el fotopigmento eritropsina.
- Cono M: ondas medias de unos 530 nm correspondiente al verde y mediante la cloropsina.
- Cono S: ondas cortas (del inglés short) de unos 430 nm correspondiente al azul y mediante la cianopsina.

Esta actividad retiniana ya es cerebral, puesto que los fotorreceptores, aunque simples, son células neuronales. La información de los conos y bastones es procesada por otras células situadas inmediatamente a continuación y conectadas detrás de ellos (horizontales, bipolares, amacrinas y ganglionares). El procesamiento en estas células es el origen de dos dimensiones o canales de pares antagónicos cromáticos: Rojo -Verde - Azul - Amarillo y de una dimensión acromática o canal de claroscuro. Dicho de otra manera, estas células se excitan o inhiben ante la mayor intensidad de la señal del Rojo frente al Verde y del Azul frente a la suma de Rojo y Verde, generando además un trayecto acromático de información relativa a la luminosidad.

III.2. Síntesis del color

Los colores primarios dependen de la fuente del color, ya que puede ser una fuente luminosa que emite una luz con un color determinado o puede tratarse de un objeto que absorbe una parte y refleja otra de la luz que recibe y que es lo que vemos e interpretamos. Tomando en cuenta estas dos fuentes de color, se puede resumir los modelos más difundidos para la síntesis del color del siguiente modo:

Tipo de síntesis	Síntesis aditiva	Síntesis sustractiva	Coloración tradicional
Fundamento	Las luces de color que se superponen, se adicionan formando tonos más claros.	Los pigmentos al mezclarse, sustraen o absorben más luz formando tonos más oscuros.	Mezcla de pigmentos que también es sustractiva, pero de naturaleza artística, tradicional y empírica.
Modelos	RGB, HSV y otros	CMY, CMYK	RYB
Colores primarios	Rojo, verde y azúl	Cian, magenta y amarillo	Azul, rojo y amarillo
Colores secunadios	Cian, magenta y amarillo	Rojo, verde y azúl	Verde, naranja y púrpura
Uso común	Monitores y proyectores.	Impressión, fotografia.	Actes pictóricas tradicionales.
			
Ejemplo	Focos luminosos	Uso de tintas	Lápiz pastel

III.2.1. La síntesis aditiva

Se llama síntesis aditiva a obtener un color de luz determinado por la suma de otros colores. El proceso de reproducción aditiva normalmente utiliza luz roja, verde y azul para producir el resto de los colores. Combinando uno de estos colores primarios con otro en proporciones iguales produce los colores aditivos secundarios, más claros que los anteriores: cian, magenta y amarillo. Variando la intensidad de cada luz de color finalmente deja ver el espectro completo de estas tres luces. La ausencia de los tres da el negro, y la suma de los tres da el blanco. Estos tres colores se corresponden con los tres picos de sensibilidad de los tres sensores de color en nuestros ojos.

Las televisiones, los monitores de ordenador y las pantallas de los teléfonos celulares, son las aplicaciones prácticas más comunes de la síntesis aditiva.

III.3. Propiedades del color

Las propiedades o cualidades fundamentales del color son tres:

- Matiz, tono o tonalidad: ubicación del color en el círculo cromático en función de las longitudes de onda.
- Saturación, colorido o pureza: distancia entre un color y su traducción en la escala de grises. A más saturación, el color será más vívido puro o colorido.
- Brillo. Según la clasificación depende de la luminosidad o el valor. La diferencia depende de la claridad u oscuridad del color. El valor mínimo corresponde al cero y el máximo depende si se trata de un modelo HSL (luminosidad) o HSV (valor).

Estas 3 propiedades combinadas entre sí, son capaces de sintetizar toda la gama de colores existente, de una forma diferente al de la combinación de los colores primarios aditivos (RGB). Esto constituye la base de la síntesis del color de los modelos HSL y HSV.

Es importante la elección de una selección adecuada de los colores utilizados.

Esto es debido a que la percepción del ser humano es subjetiva (todos conocemos algún caso de daltonismo en mayor o menor grado) y muy compleja.

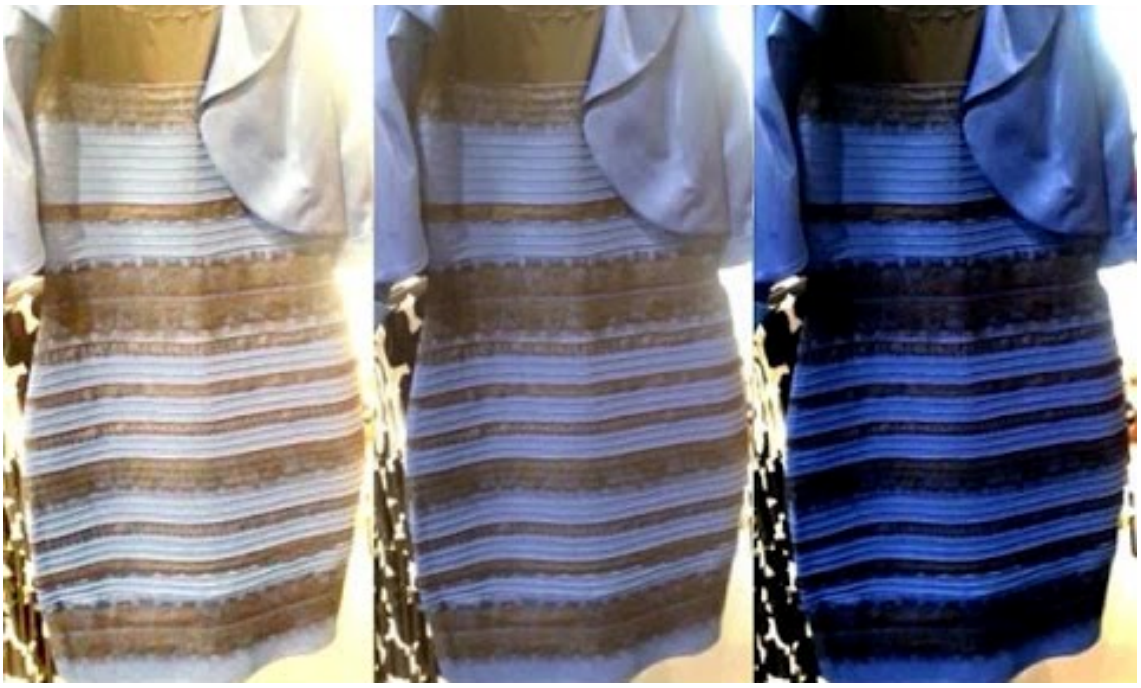


Figura 25 - Interpretación cognitiva de la imagen conflictiva debido a su elevada saturación de luminosidad.

La figura 25 es una demostración famosa de tales condiciones. El vestido originario es de color azul y negro. El cerebro filtra intuitivamente el fondo y la luz para ver el verdadero color del vestido, pero la tonalidad azulina de la imagen y la confusión de los colores que hay alrededor del vestido llevan al

cerebro a una confusión. Al ser la luz captada por el ojo, el cerebro debe asignar el color correctamente y, para ello, intenta aplicar unos filtros que corrijan la imagen fotográfica para acercarla lo máximo posible a la real. Por ejemplo, si vemos una foto en la que toda la imagen está iluminada por una luz roja y alguien viste una camiseta blanca, nuestro cerebro, automáticamente, intenta compensar ese exceso de rojo para llegar a asignar la tonalidad que más se corresponda con los colores reales y, a pesar de que en la imagen la camiseta será rojiza, interpretaremos correctamente que es en realidad de color blanco.

En el caso del vestido de la polémica, el cerebro de quienes lo ven blanco y dorado interpreta, sin que se pueda hacer nada por cambiarlo, que el vestido está a contraluz, a diferencia del fondo, lo que implica que está más oscuro, más tenue y, por contraste con el fondo amarillento, con colores más fríos (azulados), por lo que intenta compensarlo aplicando los filtros contrarios: automáticamente ilumina el vestido (lo que hace que el azul se vuelva más blanquecino y el negro, menos negro), lo satura de color (que lo que hace es realzar la iluminación amarillenta que tiene toda la imagen y crea el efecto dorado) y da más calidez a toda la fotografía (que implica aplicar más amarillo, con lo que el efecto dorado se intensifica).

El cerebro de quienes lo perciben azul y negro, sin embargo, acaba aplicando precisamente los filtros contrarios (de ahí la disparidad tan alarmante). Como interpreta que, en realidad, no es que el vestido está a contraluz, sino que toda la imagen está muy iluminada por una luz amarilla, intenta compensarlo de una forma muy distinta a la anterior: oscureciendo el vestido (el azul se vuelve más oscuro y, por tanto, más visible y el negro más cercano al negro absoluto) y compensando la calidez de la luz amarilla que lo baña todo con un filtro frío, es decir, más azulado (que realza todavía más el verdadero color del vestido). Y lo que consigue es que, finalmente, el cerebro llega a una conclusión mucho más exacta de la realidad que quienes lo ven blanco y dorado.

Por ese motivo, deberemos trabajar siempre mediante una paleta de colores claramente diferenciados que garanticen su diferenciación bajo cualquier circunstancia.

III.4. Interpretación emocional de los colores

La psicología del color es un campo dirigido a analizar el efecto del color en la percepción y la conducta humana. Esta corriente todavía es una ciencia inmadura en la corriente principal de la psicología contemporánea, teniendo en cuenta que muchas técnicas adscritas a este campo pueden categorizarse dentro del ámbito de la medicina alternativa.

A pesar de esto, pareció interesante la utilización de esta técnica en la elección de la paleta con la finalidad de no materializar una incoherencia comunicativa a la hora de la elección de los colores durante la presentación de la información. La tabla siguiente es una síntesis sobre el significado y la interpretación de los colores en el ser humano según la psicología del color y la cromoterapia.

Verde	frescura, calma, lealtad, amabilidad, consciencia, abundancia, seguridad, tranquilidad.
Azul	tranquilidad, apacibilidad, calma, orden, armonía, fiabilidad, serenidad.
Violeta	Riqueza, nobleza, creatividad, excentricidad, singularidad.
Amarillo	Alegría, gozo, felicidad, emoción, creatividad.
Naranja	Sociabilidad, exótico, cálido, picante, juguetón.
Rojo	Dramaticidad, energía, optimismo, valor, peligro, determinación, pasión.
Rosa	Feminidad, afecto, cariño, suavidad, apacibilidad, individualidad, compasividad.
Gris	Cuidado, moderno, centrado, articulado, modestia, futurista.
Negro	Potente, sofisticado, artístico, misterioso, sensible, meticuloso
Marrón	Estable, confiable, conectado, terrenal, orgánico, natural.
Blanco	Puro, inocente, bondadoso, limpio, estéril, tranquilidad, serenidad.

Al trabajar con datos de red y con tal de utilizar la psicología del color en la interpretación de los valores, crearemos una paleta que refleje los resultados mediante una congruencia comunicativa utilizando esta codificación.

III.5. Los colores Web

No todos los colores perceptibles al ojo humano pueden ser reproducidos digitalmente. El círculo cromático visual crea un dominio frecuencial continuo con infinitos valores y su traducción al mundo digital dependerá de la cantidad de bits necesarios para su representación. Este concepto es lo que denominamos profundidad del color. Debido a la naturaleza binaria del mundo binario, una profundidad de bits 'n' implica que cada unidad de píxel de la imagen puede tener 2^n posibles valores o colores distintos. Debido a la aceptación universal de los octetos (byte) como unidades básicas de información en los dispositivos digitales de almacenamiento, los valores de profundidad de color acostumbran a ser divisores o múltiplos de 8, a saber 1, 2, 4, 8, 16, 32, etc.

Los colores web son aquellos que aparecen en las páginas web. Hace algunos años, cuando los ordenadores soportaban tan solo 256 colores distintos, W3C[1] sugirió una lista de 216 Colores web seguros en su estándar, reservando 40 para colores del sistema. La paleta de los colores seguros fue creada con el propósito de asegurar la universalidad de presentación de colores en caso de utilizar una paleta de 256 colores en cualquier monitor. Esto no es importante en la actualidad ya que los colores en un monitor moderno permite la visualización de 16384 colores distintos, o lo que es lo mismo, una profundidad de color con factor 14.

[1] W3C (World Wide Web Consortium): Comunidad internacional que desarrolla estándares que aseguran el crecimiento de la web a largo plazo.

Los distintos formatos de color capaz de declarar en la web se resumen en los siguientes estándares:

- Colores Hexadecimales
- Colores RGB
- Colores RGBA
- Colores HSL
- Colores HSLA
- Colores universales o predefinidos.

Type	Definition	Explanation	Nº colors
HEX	#RRGGBB	0 to F	16.581.375
RGB	rgb(r,g,b)	0 to 255	16.581.375
RGBA	rgba(r,g,b,a)	0 to 255 (r,g,b) and 0 to 1 (a)	16.581.375 + alpha
HSL	hsl(m,s,b)	0 to 360 (m), 0 tp 100% (s, b)	3.600.000
HSLA	hsl(m,s,b,a)	0 to 360 (m), 0 tp 100% (s, b), 0 to 1 (a)	3.600.000 + alpha
Universal	Color definiton[44]		140

Para el desarrollo del proyecto he utilizado las definiciones de colores que habilitan mayor cantidad de colores posibles, es decir, la nomenclatura RGB y Hexadecimal, además de la RGBA en caso de necesitar el parámetro “alpha” para conseguir transparencias o rebajar intensidades.

III.6. La paleta elegida

Teniendo en cuenta todos los condicionantes anteriores y pretendiendo crear una paleta de colores que focalice la atención del analizador de los datos en aquellos valores sensibles de atención o acción realizamos pruebas con distintos modelos de paleta de colores. Como punto de partida elegí el rojo puro con la intención de destacar los valores mayores o de mayor relevancia.

En primer lugar pensé en la creación de una paleta de colores mediante la utilización de distintas técnicas de combinación de color. Aplicamos las técnicas de generación de colores complementarios, análogos, tríada, cuadrado, analogía, rectángulo y escisión del complementario. Esto nos generaba una escala de 12 colores posibles.

	HEX	RGB	gap
	FF7F00	255,127,0	0,+127,0
	FFFF00	255,255,0	0,+128,0
	80FF00	128,255,0	-128,0,0
	00FF00	0,255,0	-127,0,0
	00FF80	0,255,128	0,0,+128
	00FFFF	0,255,255	0,0,+127
	0080FF	0,128,255	0,-127,0

	0000FF	0,0,255	0,-128,0
	7F00FF	127,0,255	+127,0,0
	FF00FF	255,0,255	+128,0,0
	FF0080	255,0,128	0,0,-127
	FF0000	255,0,0	0,0,-128

El resultado de la aplicación de esta paleta en la aplicación comportaba una serie de ventajas: amplia distinción de los diferentes estados de cada valor, viveza de los resultados, armonía cromática, etc. Por el contrario, comportaba unas importantes limitaciones que nos hicieron descartarla:

- Reducido número de colores para la representación de valores.
- Incongruencia comunicativa: algunos colores que destacaban sobre los demás representaban datos de poca relevancia.
- Vaga diferenciación del ser humano para la sensación de variación del espectro de color verde.

La Figura 26 refleja estos resultados:

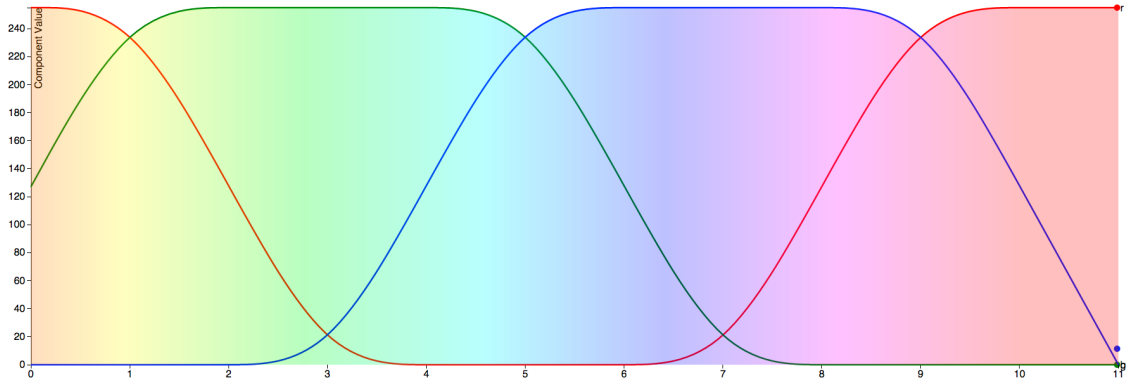


Figura 26 – Paleta de colores inicial.

Teniendo en cuenta la prueba anterior elaboramos una segunda paleta que reuniera las siguientes condiciones:

- Dar mayor prioridad a la congruencia comunicativa en lugar de la base matemática.
  - 0% – cian: información poco relevante y describe tranquilidad, armonía.
  - 50% – turquesa: información sensible a considerarse. Describe naturalidad.
  - 75% – amarillo: información a tener en cuenta. Describe alerta.
  - 90% – rojo: información muy importante. Describe peligro.
- Utilizar un mayor número de colores en la paleta, consiguiendo mayor precisión
- Innecesaria utilización de todo el espacio visual del color (Violeta, rosa, marrón y salmón descartado).

Asumiendo las siguientes condiciones realizamos una propuesta con 21 colores, prácticamente el doble, utilizando las siguientes muestras:

	00DCFF	0,220,255	0%
	00DCF3	0,220,243	5%
	00DCE7	0,220,231	10%
	00DCDC	0,220,220	15%
	00E1DC	0,225,220	20%
	00E6DC	0,230,220	25%
	00EBDC	0,235,220	30%
	00F0DC	0,240,220	35%
	00F5DC	0,245,220	40%
	00FADC	0,250,220	45%
	00FFDC	0,255,220	50%
	20FFA8	32,255,168	55%
	40FF74	64,255,116	60%
	80FF40	128,255,64	65%
	C0FF0C	192,255,12	70%
	FFFF00	255,255,0	75%
	FFD200	255,210,0	80%
	FFAA00	255,170,0	85%
	FF8200	255,130,0	90%
	FF5A00	255,90,0	95%
	FF3200	255,50,0	100%

Lo que producía la paleta continua de colores de la figura 27:

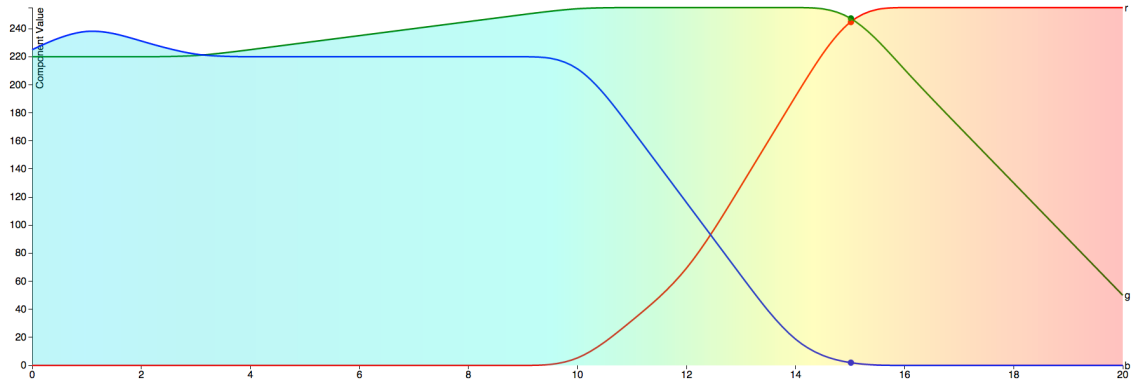


Figura 27 - Paleta de colores rectificada, partiendo de requisitos emocionales.

Tal y como queda reflejado en la figura 27, la utilización de esta paleta es mucho más acertada a la hora de comprender los resultados. A pesar de ello tiene las siguientes carencias:

- Poca o escasa diferenciación de los valores comprendidos entre el 0 y el 50%.
- Escasa utilización del verde para reflejar naturalidad en los resultados.

Teniendo en cuenta todos los resultados anteriores se realizó una serie de mejoras:

- Introducción de los colores blanco y negro.
    - Blanco: ausencia de datos.
    - Negro: Caída o fallo del sistema, catástrofe, etc.
- Mayor diferenciación entre colores que representen valores inferiores al 50%.

Teniendo en cuenta estos aspectos realice una evolución del sistema anterior introduciendo las siguientes muestras:

	FFFFFF	255,255,255	0%
	C0FFFF	192,255,255	4,76%
	00FFFF	0,255,255	9,52%
	00CDFF	0,205,255	14,29%
	00B9FF	0,185,255	19,05%
	00A5FF	0,165,255	23,81%
	0091FF	0,145,255	28,57%
	07DFF	0,125,255	33,33%
	0096D2	0,150,210	38,1%
	00C8A5	0,200,165	42,86%
	00FA78	0,250,120	47,62%
	00FF4B	0,255,75	52,38%
	20FF39	32,255,57	57,14%
	40FF27	64,255,39	61,90%
	80FF15	128,255,21	66,67%
	C0FF03	192,255,3	71,43%
	FFFF00	255,255,0	76,19%
	FFD200	255,210,0	80,95%
	FFAA00	255,170,0	85,71%
	FF8200	255,130,0	90,48%
	FF5A	255,90,0	95,24%
	FF3200	255,50,0	100%

El negro tiene un problema al añadirlo. Todos los colores se utilizan con una opacidad. Cómo el fondo utilizado por la aplicación es claro, el negro se transforma en un gris muy tenue, de forma que en caso de querer representar la superación del máximo permitido, utilizaremos el color negro sin degradar y forzado por código. Po este motivo no lo introducimos en la paleta de colores.

Al utilizar estas muestras el resultado obtenido es el siguiente (Ver figura 28):

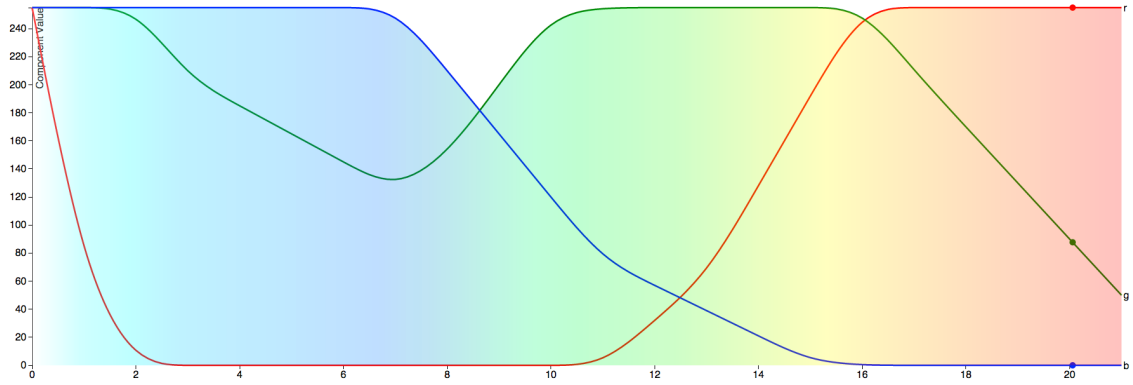


Figura 28 - Paleta de colores definitiva.

ANEXO IV: LA BASE ESTRUCTURA DE LA BASE DE DATOS DE LA APLICACIÓN

IV.1. Almacenamiento de la configuración de la herramienta

En primer lugar, destacamos la tabla “options”, que se encarga de gestionar la configuración de las opciones y características de la aplicación. En ella se registran todos aquellos parámetros personalizables necesarios durante el natural transcurso de la utilización de la aplicación desde el momento que se encuentra instalada.

id	int(11)	Único, índice, Auto-incremental
name	varchar(100)	
value	text	

Este patrón de tabla se ha extraído del modelo de diferentes CMS para almacenar este tipo de información. Permite un almacenamiento variado de forma que puede albergar opciones de todo tipo y longitud.

Tras su creación introducimos en ella los valores de configuración, cómo son el driver SQL de la conexión (MySQL - MySQLi), la IP del Host, nombre de usuario, contraseña y nombre de la base de datos, nombre de usuario del administrador de la aplicación y su contraseña de acceso cifrada, nombre de la aplicación, descripción del sitio y directorio URL de la instalación.

IV.2. Almacenamiento de usuarios

Además, incluye una tabla “users”, encargada del almacenamiento de la información de los usuarios de la aplicación. Los campos de ésta son

Id	Int(11)	Clave primaria, auto-incremental
Username	Varchar(50)	Único
Email	Varchar(50)	
Password	Varchar(50)	
User_type	Varchar(10)	
Activation_key	Varchar(50)	
Activation_state	Tinyint(1)	
Social_id	Varchar(50)	
Social_type	Varchar(10)	
Avatar	Varchar(150)	
Fullname	Varchar(50)	
phone	Varchar(20)	
location	Varchar(20)	
About	Varchar(150)	
gender	Varchar(1)	
Join_date	timestamp	

Esta estructura de base de datos está extraída a partir de la consulta de las distintas propuestas y alternativas existentes e integra además del registro clásico, unos campos adicionales para el registro mediante redes sociales. Los campos de activación son necesarios para, en caso del registro clásico, el proceso de conformación de usuario mediante correo. El campo "user_type" almacena la condición del usuario que determinará sus permisos de acceso en el aplicativo.

IV.3. Almacenamiento de datos genéricos

Incluye también dos tablas que actúan de índice de elementos: la de nodos, más compleja, y la de enlaces además de una tabla denominada datastore, dónde se concentrarán cualquiera de los datos de estudio más complejos, cómo pueden ser tráficos, encaminamientos, combinaciones de estos u otro tipo.

En la tabla nodos, que actúa como índice de cada uno de estos elementos encontramos los distintos atributos:

Id	Int(11)	PK, Auto-incr.	
Name	Varchar(100)	Único	Hash del nombre que actúa de identificador
Fullname	Varchar(100)		Nombre completo del nodo
Description	Varchar(250)		Descripción del nodo
depth	Int(11)		Profundidad en la navegación
type	Varchar(50)		Tipo de nodo
lat	Float		Coordenada geográfica de latitud
lon	Float		Coordenada geográfica de longitud
up	Datetime		Fecha de inserción en la BBDD
Margin_right	Int(11)		Estructura jerárquica
Margin_left	Int(11)		Estructura jerárquica

Esta tabla tiene algunas particularidades en el uso y que necesitan una aclaración. En primer lugar, el campo name es un identificador único y almacena el hash crc32[2] del fullname. El resultado es un string de 8 caracteres de longitud. El campo depth es un limitador del máximo nivel de profundidad visible para el nodo. Esto significa que cualquier nodo con valor depth comprendido entre el valor depth del padre inmediato en la jerarquía y el suyo propio será visible y en caso contrario, estará filtrado. Por último, en el caso de los "margins" (right y left), actúan en conjunto como selectores de jerarquía. Para la comprensión de este funcionamiento, consultar en los anexos el apartado *La gestión de datos jerárquicos en MySQL*.

Por motivos de requerimientos a nivel de front-end, existe una tabla que almacena los tipos de nodos que existen como índice, para que funcionen a semejanza que etiquetas, mediante sugerencias.

[2] **crc32**: Verificación por redundancia cíclica generalmente utilizado para la detección de errores utilizado frecuentemente en redes digitales y dispositivos de almacenamiento para detectar cambios accidentales en los datos.

La tabla de almacenamiento de enlaces, por el contrario, es más simple. Ésta tan sólo alberga relaciones entre nodos ya existentes de manera que la estructura es la siguiente:

Id	Int(11)	PK, Único, Autoincrement	
Source	Varchar(100)		Id nodo origen
Target	Varchar(100)		Id nodo destino
Description	Varchar(250)		Descripción del nodo
Capacity	Int(11)		Bps del enlace
up	datetime		Fecha de inclusión

Como podemos apreciar, esta tabla se sirve de la anterior (tabla de nodos) para almacenar los enlaces además de albergar algunos detalles técnicos propios del enlace como puede ser la velocidad de transmisión máxima que permite y una breve descripción a modo informativo.

Por último, dentro de esta categoría de tablas encontramos la denominada “datastore”. Esta tabla funciona a modo de contenedor de datos de cualquier tipo. La define la siguiente estructura:

Title	Varchar(50)	OK	Nombre de la var
Data	Mediumtext		datos
unserialize	Smallint(5)	NOT NULL, DEFAULT ’0’	Serializado/no

Queda reflejado que bajo cualquier nombre de variable podremos almacenar datos en el campo data especificando si se encuentran serializados o no. Esta tabla nos permite albergar cualquier dato distinto de lo ya reflejado (nodos y enlaces) debido a su estructura sencilla pero versátil.

IV.4. Tablas de almacenamiento RRDB

El propósito de estas tablas es almacenar información temporal de estado de diferentes elementos. El modelo debe funcionar cómo una máquina de estados en la que sólo debemos incluir la muestra discreta de la unidad de medida menor en la tabla correspondiente y mediante esta inclusión, realizar toda la lógica correspondiente a las medidas de unidades superiores respectivas. Pensando en la óptima información a proporcionar consideramos, de partida, cuál sería la ventana de visualización adecuada teniendo en cuanta las limitaciones de almacenamiento. De esta manera, tras consultar distintos modelos y realizar unas pruebas a nivel de interfaz de usuario, el criterio fue almacenar el total correspondiente a seis unidades de la unidad inmediata superior albergando muestras discretas correspondientes a una unidad de la unidad de almacenamiento respectiva. Teniendo en cuenta que almacenamos valores discretos en unidades de minuto cómo unidad menor esto se traduce en la sucesiva tabla de correspondencia:

	MINUTOS	HORAS	DIAS	MESES
TOTAL tiempo almacenado	360 min	144 horas	180 días	*
correspondencia	6 horas	6 días	6 meses	*
Muestra cada	1 minuto	1 hora	1 día	30 días
Δ(t) en milisegundos	60K	1.44M	86,4M	2.592M

Destacamos los asteriscos de la tabla anterior. A modo aclaratorio cabe decir que en este caso, calculando lo que supone almacenar muestras cada 30 días correspondientes a la media de las muestras mensuales, entendemos que limitar esta tabla no es conveniente ya que podemos imaginar que en caso de pretender almacenar un valor de 500 muestras (lo que no es una cantidad de datos importante en absoluto conociendo las capacidades de esta tecnología) correspondería con un valor total de unos 40 años, lo que resulta más que suficiente considerando la obsolescencia tecnológica imperante.

A pesar de todo, realizamos una prueba de almacenamiento en el servidor utilizado para conocer aproximadamente el límite de datos a hospedar, cuyo límite de almacenamiento permitía tan sólo 250Mb. Para el escenario, se prepararon dos elementos.

En primer lugar una tabla llamada pruebas con tres campos:

- Id: int(11)
- Value: long
- Date: datetime

Esta estructura corresponde al peor de los casos en supuesto de maximizar el número de nodos y minimizar la repartición de los enlaces entre ellos.

En segundo lugar preparamos un cron minutero (el menor intervalo permitido por el servidor) que lanzaba un script con eventos cada segundo (60 eventos). De esta forma saltábamos la limitación del servidor. Cada evento introducía en la base de datos una entrada completa. El límite de almacenamiento lo encontramos en las 4.982.025 entradas. Si cada RRD alberga en total 756 entradas (360+144+180+72) considerando que almacenamos los valores de 6 años, nos da una capacidad para almacenar 6590 elementos (nodos + enlaces) en el peor de los casos. Cabe tener en cuenta que esta limitación es tan solo dada por el servidor utilizado para el desarrollo de esta aplicación, y no una limitación intrínseca de la aplicación.

Al inicio del planteamiento de almacenamiento de este tipo de datos nos encontramos delante de dos modelos diferentes. Evidentemente, cada modelo supone ventajas e inconvenientes que reflejamos a continuación.

1. Almacenar todos los datos temporales en una base de datos única con un número de columnas resultado de cada elemento a muestrear.
    - Ventajas:
        - Clarividencia y ordenación en la lista de tablas de la base de datos
    - Inconvenientes:
        - Bloqueo de la tabla en operaciones complicadas

2. Almacenar los datos temporales en múltiples bases de datos con un número de columnas resultantes igual a los elementos a muestrear de cada tabla.
    - Ventajas:
        - Posibilidad de seccionar las tablas en distintos servidores en cloud (complicación de cálculos)
        - Bloqueo de tabla balanceado.
        - Simplicidad al trabajar sobre la tabla del nodo especificado.
        - Menor riesgo de eliminación y creación de tablas.
        - Mayor simplicidad en la creación del disparador de eventos Round Robin.
    - Inconvenientes:
        - Complicación a la hora de navegar por las tablas a nivel de usuario

Teniendo en cuenta todas estas cuestiones, parece evidente, entonces escoger la segunda opción debido a que a nivel de uso supone importantes ventajas; sobre todo en cuanto a la cuestión del bloqueo de las tablas.

De esta manera, dividimos las tablas de elementos Round Robin en diferentes tablas. Minimizando el número de éstas agruparemos los elementos a cuantificar en rrd agrupándolas por nodos. Teniendo en cuanta que los elementos a cuantificar son el throughput del nodo y los enlaces (%) agruparemos la información por nombre de nodo y cada tabla contendrá el throughput del propio nodo y el de los enlaces cuya fuente sea este mismo, de manera que la estructura de las tablas rrd será dinámica. Para tenerlas listadas de manera continua utilizaremos el prefijo 'rrd_' seguidas por el campo 'name' del nodo y la unidad de medida que almacena (min, hour, day, month). Esto da como resultado un nombre de la siguiente forma: rrd_[node.name]_[time.ud]. Para el índice de la tabla respectiva utilizaremos la misma forma con el sufijo '_key' dando como resultado rrd_[node.name]_[time.ud]_key.

Así, tendremos las tablas:

rrd_[node.name]_min
rrd_[node.name]_hour
rrd_[node.name]_day
rrd_[node.name]_month

rrd_[node.name]_min_key
rrd_[node.name]_hour_key
rrd_[node.name]_day_key

La estructura de estas tablas es dinámica, y debe contener el valor de los elementos a cuantificar temporalmente además de los propios para el funcionamiento de la rrdb. Al agrupar por nodos los elementos podríamos considerar que con nombrar las columnas de los enlaces con el valor del destino sería suficiente. Sin embargo, teniendo en cuanta que puede haber duplicidad de enlaces (mismo origen y destino) esto no resulta un elemento factible. Para asegurar un nombre de columna único utilizaremos imprescindiblemente un elemento que sí que es único para los enlaces: su identificador. Cómo también queremos que sea amigable para el usuario, combinaremos el identificador (imprescindible) con el valor que corresponde al identificador de la fuente. Esta configuración nos da cómo resultado la siguiente nomenclatura para las columnas de enlaces: [node_id]_[link_id]. (Figura 29)

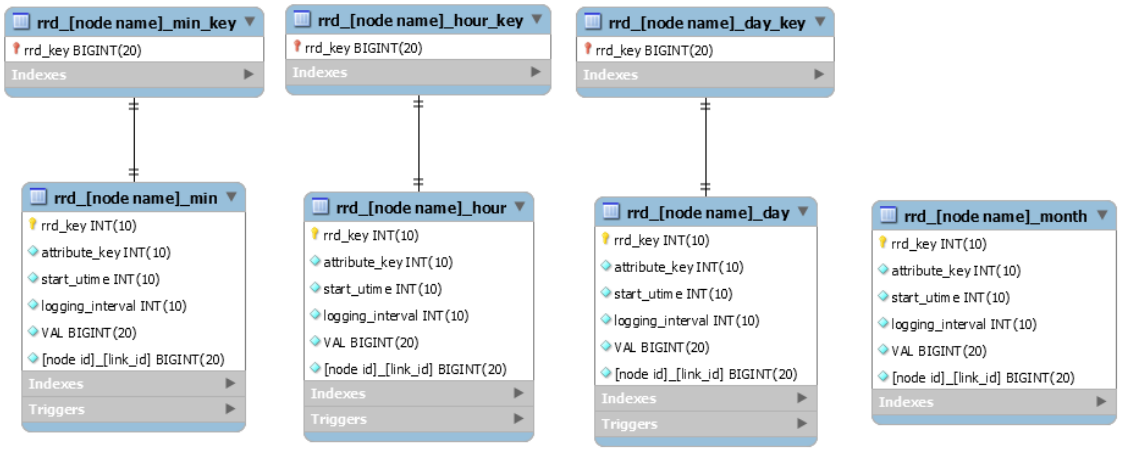


Figura 29 – Estructura de tablas rrd de los elementos que agrupa cada nodo.

En la figura 29 cabe destacar unos detalles. En primer lugar, ésta es la estructura imprescindible de tablas para cada nodo existente en la tabla nodos. En segundo lugar, los nombres de las tablas tienen un campo [node_name] que encontramos en la tabla que almacena su información detallada. En tercer lugar, precisar que cada tabla de almacenamiento de unidad tiene además un disparador que efectúa la inserción en la tabla de almacenamiento superior. En cuarto lugar, las tablas que actúan como puntero de inserción de las tablas de unidad albergan tan solo un valor que realiza esta función y el disparador de la tabla de almacenamiento de esa unidad va reconfigurando. En quinto y último lugar, cada tabla de almacenamiento de unidad de información contiene las columnas de enlaces y este número de columnas es dinámico. En este caso hemos indicado tan solo la fórmula en que aparecería el nombre de cada enlace cuyo origen partiera del nodo propio mediante sus disparadores.

IV.4.1. Funcionamiento de las tablas RRDB

La inserción de valores en esta estructura se debe hacer únicamente en la tabla con unidad de almacenamiento menor, en este caso la minutera, y con la frecuencia precisada por ésta (un valor por minuto). Al realizar esta inserción, la tabla actúa cómo una máquina de estados realizando todas las operaciones pertinentes preparando la próxima inserción e insertando, en caso necesario, los valores de las tablas de unidad mayor tal y como muestra la figura 30.

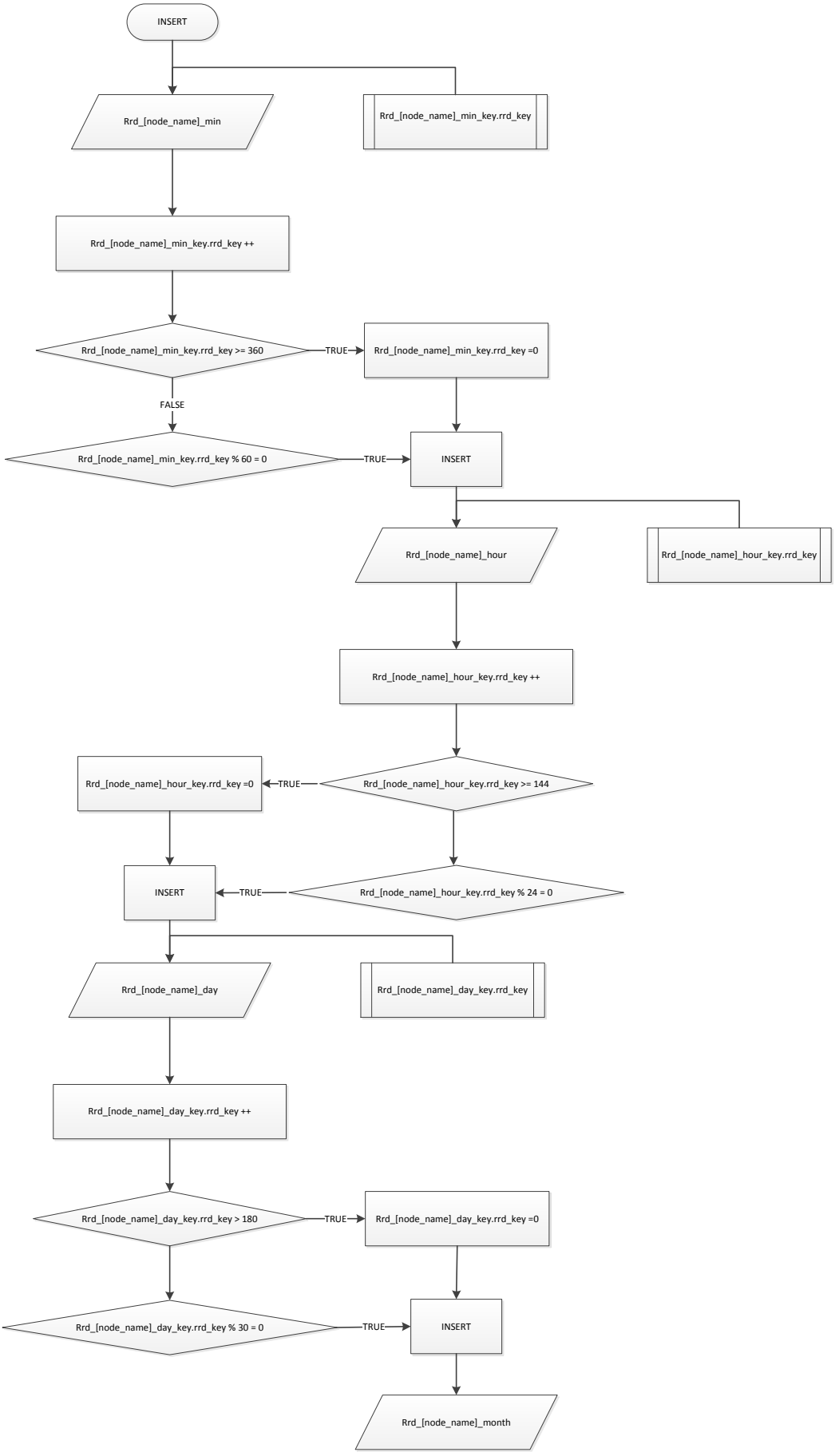


Figura 30 - Lógica de inserción de valores en la estructura rrd de las tablas de nodo.

En el apartado “Funcionamiento de las tablas Round Robin” del anexo VIII detallo en profundidad el proceso de creación, inserción y actualización de las mismas.

IV.4.2. Almacenamiento de eventos del disparador

Esta tabla es la que acomete el cron del servidor para conocer el conjunto de instrucciones de inserción que debe realizar. Tiene la siguiente estructura:

Id	Int(11)	PK, NOT NULL, ÚNICO	
Id_node	Int(11)		Identificador del nodo
Id_name	Varchar(8)		Nombre del nodo (crc(8))

Cualquier nodo que se encuentre referenciado en esta tabla y además cuente con su estructura de tablas RRD en la base de datos, obtendrá el conjunto de valores específicos necesarios para su actualización.

ANEXO V: LOS SKETCHES DE LA APLICACIÓN

V.1. Las secciones de la aplicación web

Existen dos secciones en el proyecto: sección interna y sección externa. Su presentación depende únicamente de la existencia del establecimiento de una sesión por parte del usuario (Figura 31 y 32).

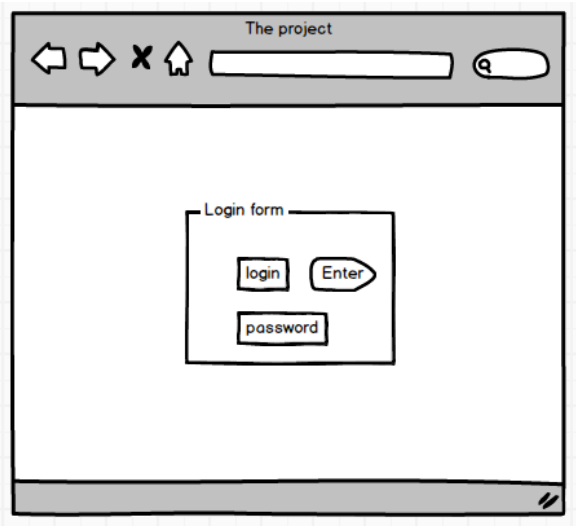


Figura 31 – Mockup de la sección externa. Página de acceso a la aplicación.

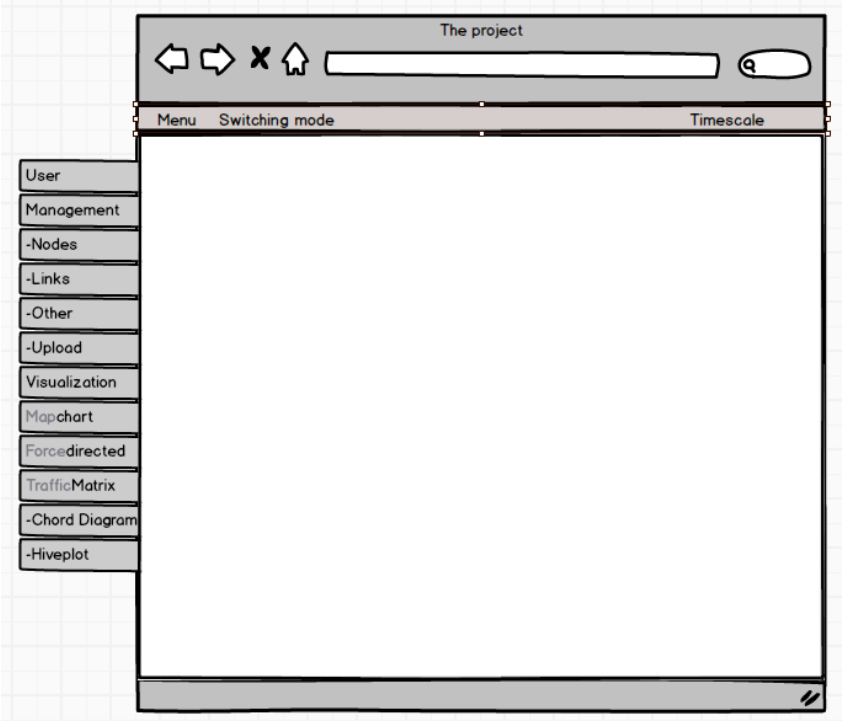


Figura 32 – Mockup de la sección interna. En la parte superior incluye un menú fijo siempre presente. A la derecha se presenta el menú desplegable. Las funcionalidades de éste dependen de los permisos del usuario.

V.2. La navegabilidad de la sección interna

La sección interna contiene dos sub-apartados:

- Modo edición y manipulación datos.
- Modo visualización de datos

Su habilitación depende de los permisos de cada usuario y sus perfiles. El bloque que contiene los niveles de navegabilidad de visualización es común para cualquier usuario registrado en la plataforma a diferencia del de manipulación y edición, que sólo tiene permitido su acceso a los usuarios de perfil administrador.

V.2.1. Modo edición y manipulación de elementos

Este modo habilita opciones para la integración, creación, manipulación edición, reemplazo y eliminación de cualquier modelo de datos. Consta de dos plantillas principales: la de manipulación de objetos de estudio y la de carga de archivos de datos. (ver figura 33, 34)

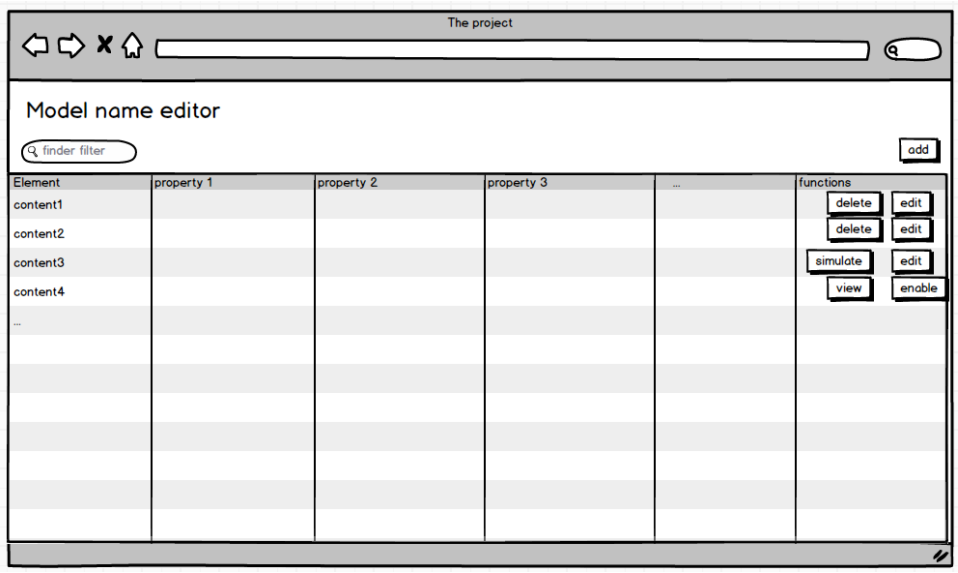


Figura 33 – Mockup de manipulación de elementos.

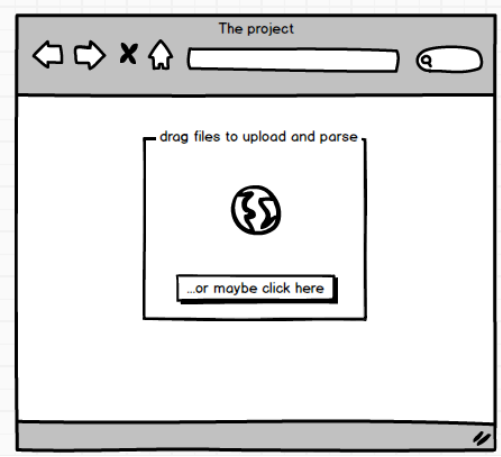


Figura 34 – Mockup del "uploader" o importador de archivos de datos.

Durante una de las iteraciones del proceso de desarrollo explicado en el apartado 4.3 de la memoria encontramos un vacío en el borrador inicial. La aplicación debía dar cabida a una ordenación jerárquica de elementos y las soluciones existentes no resultaban amigables ni efectivas (y mucho menos para una plataforma móvil). Por este motivo hubo que idear una solución nueva. (Ver figura 35)

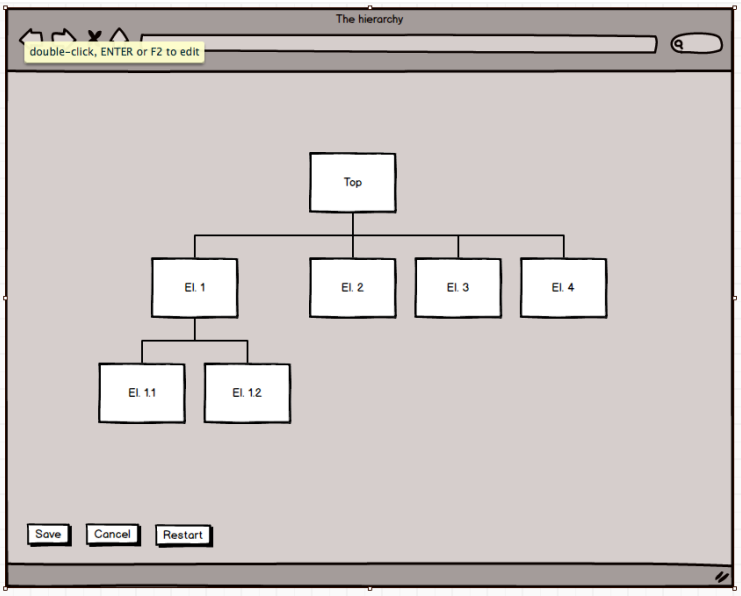


Figura 35 – Mockup del visualizador jerárquico de los elementos.

V.2.2. Modo visualización

Este modo de operación de la aplicación es accesible por cualquier usuario registrado en la plataforma. En él se debe mostrar al usuario de una forma agradable todos los datos sujetos a estudio.

Partimos de una idea conceptual de unos cuantos visualizadores ya existentes y elaboramos nuestra propuesta en mockup, reflejada en las figuras 36, 37, 38 y 39.

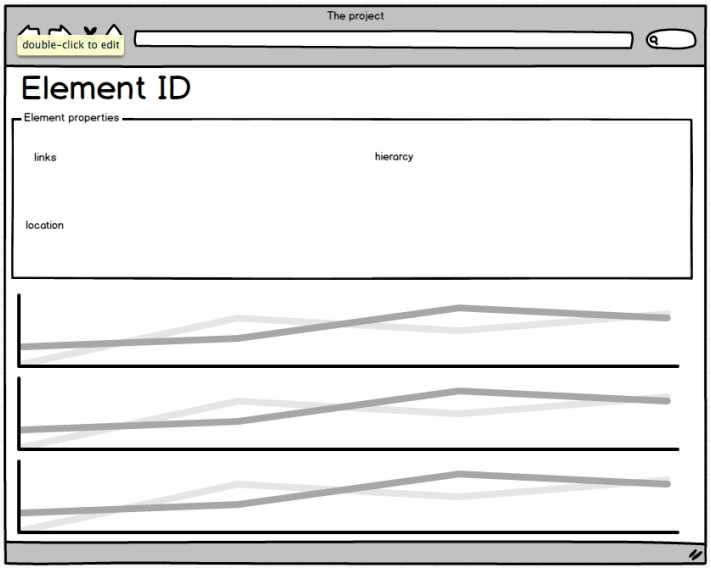


Figura 36 – Mockup del visualizador de la evolución temporal de los elementos en diferentes escalas de tiempo a semejanza de MRTG.

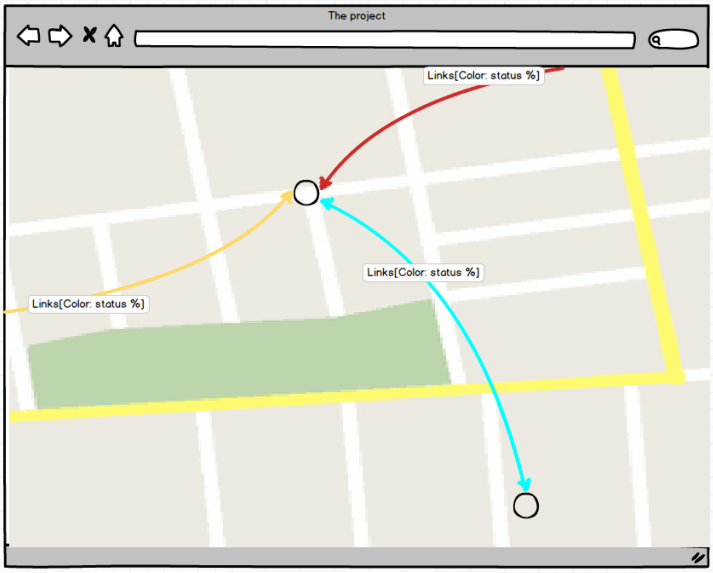


Figura 37 – Mockup del visualizador topográfico de los elementos nodo y sus enlaces.

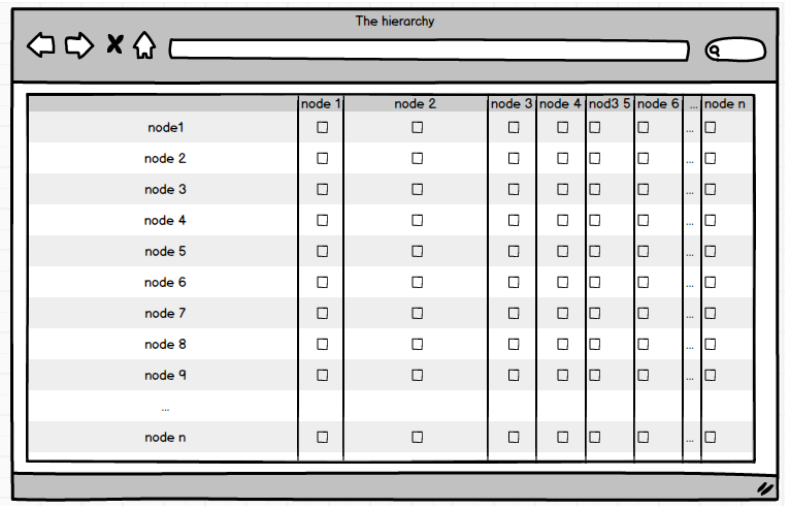


Figura 38 – Mockup del visualizador de la matriz de tráfico.

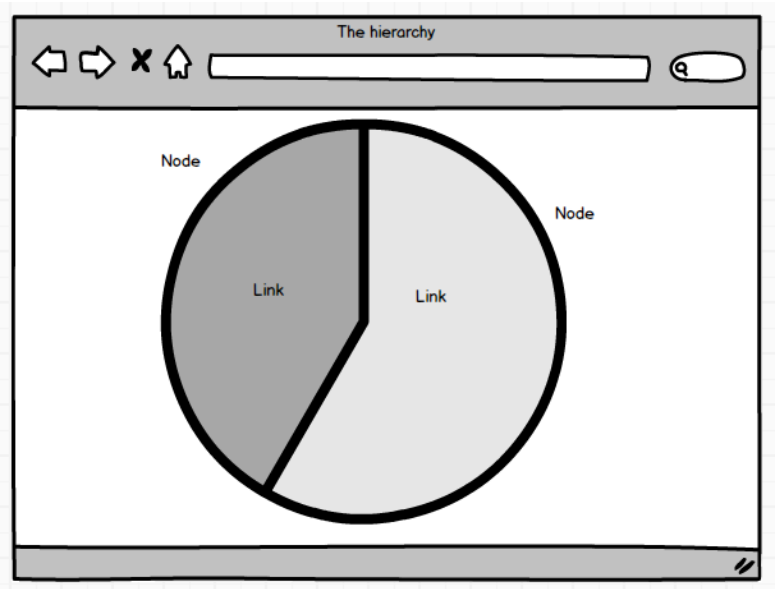


Figura 39 – Mockup de una nueva propuesta de visualización de la matriz de tráfico. Para ella se utilizará el modelo de visualización de los chord-diagrams.

Durante el desarrollo de la programación y siguiendo el proceso descrito en el apartado 4.3 de la memoria, encontramos interesante realizar una pequeña variación que aportara múltiples ventajas a estas visualizaciones. Entre ellas destacamos:

- Posibilidad de incorporar una cortina de tiempo y ajustar la escala visible.
- Duplicidad de la información en ambos modelos para facilitar la interpretación según el modelo que sea más cómodo al usuario.
- Ofrecer la mejor aportación de cada modelo al mismo tiempo.

Esta propuesta está reflejada en la figura 40.

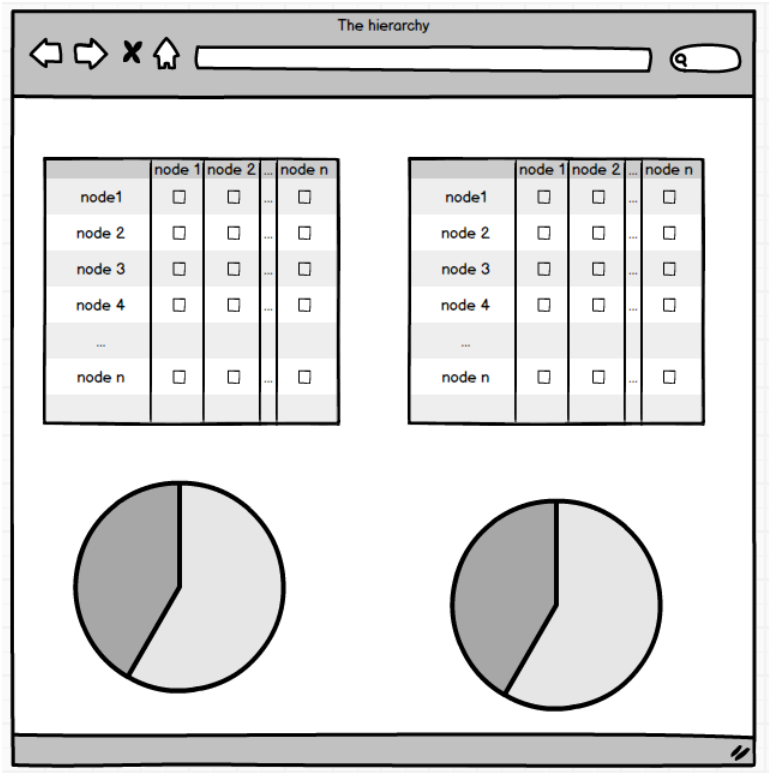


Figura 40 – Mockup de integración de los modelos de visualización de la matriz de tráfico y el chord-diagram.

ANEXO VI: EL INSTALADOR

VI.1. Instalación de la aplicación

La aplicación cuenta con un instalador sencillo. Se trata de un formulario de 13 campos y una última etapa de confirmación. Para su lanzamiento se requiere que cualquier archivo de la aplicación incluya el comprobador de la instalación y re-direccione a este módulo en caso negativo. El módulo comprobador no hace otra cosa que consultar la existencia de un archivo llamado “config.php” en el directorio principal.

Para la interfaz de instalación utilizamos una propuesta de la empresa Tympanus que ofrece de forma gratuita su libre manipulación. Ésta se encuentra en su sección codrops y lleva por título “Fullscreen Form Interface”.

Este interfaz permite de forma clarividente, sencilla y muy rápidamente completar los cómodos pasos que requiere la aplicación. Nos ecantamos por esta solución debido a que los pasos eran precisados uno por uno al rellenar el formulario, y como en el caso de la aplicación, las respuestas futuras dependen de las anteriores, resultaba uno de los más convenientes y acertados.

Cada cuestión del formulario se presenta al usuario en pantalla completa, ofreciendo a su vez la numeración de la pregunta en que se encuentra y su totalidad en la parte superior izquierda de la pantalla (Ver figura 41).

Figura 41 – Instalador de la aplicación.

En la parte derecha, a media altura encontramos a su vez alineado con la paginación 13 bullets dispuestos verticalmente en el centro de la pantalla. Cada

uno de ellos representa un apartado del instalador y tan solo permiten la navegación hacia niveles del instalador ya completados en anterioridad.

Los pasos de la instalación tienen que ver con 3 etapas:

- Datos del servidor
- Datos del administrador
- Configuración y personalización de la herramienta

VI.1.1. Datos del servidor

Corresponde a los cinco primeros pasos del instalador. En estos encontramos:

- El driver PHP del protocolo de comunicación con la base de datos. Es un selector que da a elegir entre dos opciones:
  - MySQL: driver más antiguo y extendido
  - MySQLi: driver PHP nuevo. Está orientado a objetos y tiene la característica y ventaja que permite realizar multi-consulta en la misma instrucción.
- Nombre del Host: Ip del host que hospeda la base de datos. Por defecto pondremos localhost en caso de tenerla hospedada en la misma dirección que la herramienta.
- Nombre de usuario de la base de datos: El nombre de usuario con privilegios a la base de datos de la aplicación
- Contraseña de usuario: La contraseña del usuario anterior.
- Nombre de la base de datos. El nombre de la base de datos donde se hospedará la información en la aplicación.

VI.1.2. Datos del usuario administrador

Los siguientes pasos corresponden a la información de usuario y su uso de la aplicación:

- Nombre: Nombre completo del administrador.
- Dirección de correo: EL correo de contacto del administrador y creador de la aplicación.
- Nombre de usuario: El nombre de usuario con privilegios de administrador y creador de la aplicación.
- Doble campo de creación y conformación de la contraseña del administrador para verificar la correcta introducción.

VI.1.3. Configuración y personalización de la herramienta

Ofrece la posibilidad de tres campos de personalización de la herramienta:

- Nombre de la aplicación
- Descripción
- Color de fondo.

VI.1.4. Confirmación de la instalación

Una vez completados los campos anteriores, la aplicación te ofrece en un pantallazo la visualización completa de todos éstos para confirmarlos y, en caso que fuera necesario, corregir los posibles errores que hubieran podido producirse antes del envío.

Tal y como indica la figura 42, el envío del formulario de confirmación supone la instalación de la aplicación y ninguno de estos valores podrá ser manipulado en un futuro salvo modificación en la base de datos.

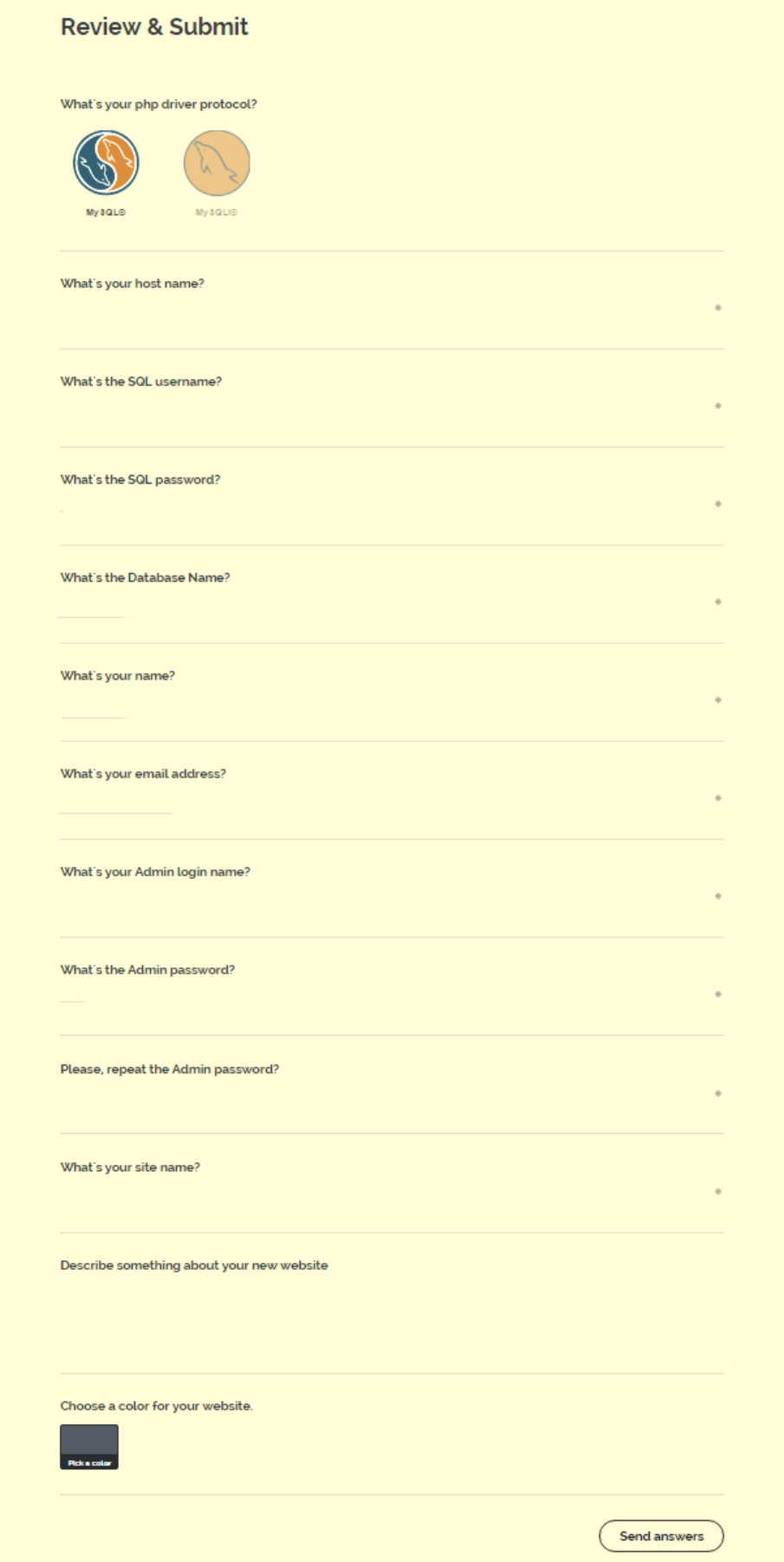


Figura 42 – Confirmación del formulario de instalación.

ANEXO VII: GESTIÓN DE DATOS JERÁRQUICOS EN MYSQL

VII.1. Introducción

La mayoría de los usuarios nos hemos encontrado alguna vez con la situación de almacenar datos jerárquicos en una base de datos y hemos podido comprobar que esta tarea no está contemplada de manera simple. Las tablas relacionales de una base de datos no son jerárquicas por naturaleza sino que son sencillamente una simple lista. De esta forma, los datos jerárquicos con relación padre-hijo no están representados de forma natural en una base de datos relacional.

Para mi propósito, los datos jerárquicos no son más que una colección de datos donde cada elemento tiene un solo padre y puede pender de él algún hijo (a excepción del elemento raíz, que no tiene padre). Los datos jerárquicos se pueden encontrar en una amplia variedad de aplicaciones de bases de datos que incluyen foros e hilos de listas de correos, organigramas de negocio, categorías de gestión de contenidos y categorías de productos. Para la demostración escogeremos un modelo sencillo de datos ficticios del tipo producto de una tienda de electrónica.

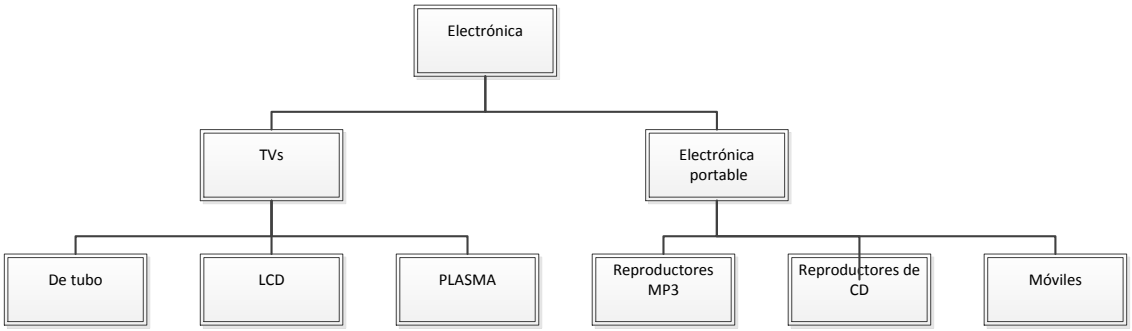


Figura 43 – Modelo simplificado del estudio.

Las categorías del ejemplo conforman una jerarquía de la misma manera que en los otros ejemplos citados anteriormente.

VII.2. El modelo de lista de adyacencia

Normalmente las categorías del ejemplo anterior se almacenarán en una tabla como la siguiente:

```
CREATE TABLE category(
        category_id INT AUTO_INCREMENT PRIMARY KEY,
        name VARCHAR(20) NOT NULL,
        parent INT DEFAULT NULL
);

INSERT INTO category VALUES(1,'ELECTRONICA',NULL),(2,'TVs',1),(3,'De TUBO',2),
```

```
        (4,'LCD',2),(5,'PLASMA',2),(6,'ELECTRONICA POSRABLE',1),(7,'Reproductores
MP3',6),(9,'Reproductores de CD',6),(10,'Móviles',6);

SELECT * FROM category ORDER BY category_id;
+-------------+----------------------+--------+
| category_id | name                 | parent |
+-------------+----------------------+--------+
|           1 | ELECTRONICA          |   NULL |
|           2 | TVS                  |      1 |
|           3 | de tubo              |      2 |
|           4 | LCD                  |      2 |
|           5 | PLASMA               |      2 |
|           6 | ELECTRÓNICA PORTABLE |      1 |
|           7 | REPRODUCTORES MP3    |      6 |
|           8 | REPRODUCTOR de CD    |      6 |
|           9 | Móviles              |      6 |
+-------------+----------------------+--------+
9 rows in set (0.00 sec)
```

En este modelo, cada elemento de la tabla contiene un puntero a su padre. El elemento superior, en este caso, la electrónica, tiene un valor NULL para su padre. Tiene como ventaja ser de comprensión muy simple. Esto quiere decir que es fácil percibir que los reproductores MP3 son hijo de electrónica portable y éste, hijo de electrónica. Mientras que este modelo puede ser más simple de manipular en el lado cliente puede resultar más complejo en caso de querer trabajar únicamente en SQL. Para el caso de trabajar con un volumen elevado de datos es imprescindible la optimización de la base de datos y las consultas a ésta, por lo que este modelo resultaría inviable.

VII.2.1. Recuperar el árbol completo

La primera tarea en cuanto a la estructura de datos jerárquica es, sencillamente, la obtención del árbol completo, generalmente mediante alguna forma de anexión. La forma más común de realizar esta tarea mediante SQL puro es un self-join:

```
SELECT t1.name AS lev1, t2.name as lev2, t3.name as lev3, t4.name as lev4
FROM category AS t1
LEFT JOIN category AS t2 ON t2.parent = t1.category_id
LEFT JOIN category AS t3 ON t3.parent = t2.category_id
LEFT JOIN category AS t4 ON t4.parent = t3.category_id
WHERE t1.name = 'ELECTRONICS';

+-------------+----------------------+----------------------+-------+
| lev1        | lev2                 | lev3                 | lev4  |
+-------------+----------------------+----------------------+-------+
| ELECTRONICA | TVS                  | De tubo              | NULL  |
| ELECTRONICA | TVS                  | LCD                  | NULL  |
| ELECTRONICA | TVS                  | PLASMA               | NULL  |
| ELECTRONICA | ELECTÓNICA PORTABLE  | Reproductores de CD  | NULL  |
| ELECTRONICA | ELECTÓNICA PORTABLE  | CD PLAYERS           | NULL  |
| ELECTRONICA | ELECTÓNICA PORTABLE  | Móviles              | NULL  |
+-------------+----------------------+----------------------+-------+
6 rows in set (0.00 sec)
```

VII.2.2. Obtener los elementos finales (hojas)

Podemos obtener los últimos nodos de nuestro árbol mediante la siguiente consulta del tipo LEFT-JOIN:

```
SELECT t1.name FROM
category AS t1 LEFT JOIN category as t2
ON t1.category_id = t2.parent
WHERE t2.category_id IS NULL;

+----------------------+
| name                 |
+----------------------+
| De tubo              |
| LCD                  |
| PLASMA               |
| Reproductores de CD  |
| Móviles              |
+----------------------+
```

VII.2.3. Obtener una rama concreta

El self-join también permite obtener la rama completa a través de las jerarquías:

```
SELECT t1.name AS lev1, t2.name as lev2, t3.name as lev3, t4.name as lev4
FROM category AS t1
LEFT JOIN category AS t2 ON t2.parent = t1.category_id
LEFT JOIN category AS t3 ON t3.parent = t2.category_id
LEFT JOIN category AS t4 ON t4.parent = t3.category_id
WHERE t1.name = 'ELECTRONICS' AND t4.name = 'FLASH';

+-------------+----------------------+-------------+-------+
| lev1        | lev2                 | lev3        | lev4  |
+-------------+----------------------+-------------+-------+
| ELECTRONICA | ELECTRÓNICA PORTABLE | MP3         | NULL  |
+-------------+----------------------+-------------+-------+
1 row in set (0.01 sec)
```

El principal condicionante de este enfoque es que es imprescindible un self-join por cada nivel de la jerarquía. El rendimiento, por tanto se ve afectado por cada nivel jerárquico, además de la implicación estricta del conocimiento del número de niveles de la jerarquía.

VII.2.4. Limitaciones del modelo

La manipulación de este modelo en el nivel SQL puede resultar complicada. Antes de poder ver la ruta completa de una jerarquía debemos conocer el nivel al que pertenece. Al eliminar los nodos es imprescindible, además, la supresión de cualquier descendiente existente, ya que, en caso contrario dejaría nodos huérfanos. Muchos de estos inconvenientes pueden abordarse mediante el uso de código en el lado cliente o mediante rutinas. Mediante código procedimental podemos empezar por los elementos finales e ir recorriendo el árbol ascendentemente y mediante este procedimiento crear rutinas como la manipulación o eliminación de ciertas ramas o un cambio tanto de rama como de nivel de jerarquía.

VII.3. El modelo de lista anidada

Este modelo, aunque de más complicada comprensión, consiste en un enfoque absolutamente diferente y se conoce comúnmente como el modelo anidado.

Este modelo requiere imaginar nuestra jerarquía de una manera nueva como contenedores anidados en lugar de nodos y enlaces tal y como demuestra la figura 44.

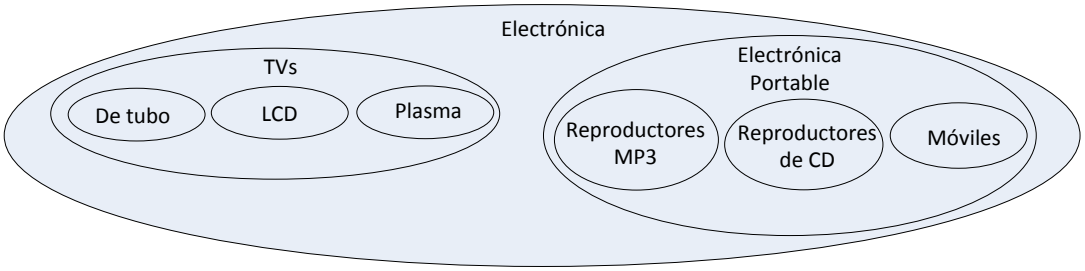


Figura 44 – Representación anidada del modelo del estudio.

Podemos apreciar cómo la jerarquía se mantiene de forma que los elementos padre envuelven sus hijos. Representamos esta forma de jerarquía mediante la utilización de los valores izquierda y derecha para la representación de la anidación de los nodos.

```
CREATE TABLE nested_category (
        category_id INT AUTO_INCREMENT PRIMARY KEY,
        name VARCHAR(20) NOT NULL,
        lft INT NOT NULL,
        rgt INT NOT NULL
);

INSERT INTO nested_category
VALUES(1,'ELECTRONICS',1,20),(2,'TELEVISIONS',2,9),(3,'TUBE',3,4),
 (4,'LCD',5,6),(5,'PLASMA',7,8),(6,'PORTABLE ELECTRONICS',10,19),(7,'MP3
PLAYERS',11,14),(8,'FLASH',12,13),
 (9,'CD PLAYERS',15,16),(10,'MÓVILES',17,18);

SELECT * FROM nested_category ORDER BY category_id;

+-------------+----------------------+-----+-----+
| category_id | name                 | lft | rgt |
+-------------+----------------------+-----+-----+
|           1 | ELECTRONICS          |   1 |  20 |
|           2 | TELEVISIONS          |   2 |   9 |
|           3 | TUBE                 |   3 |   4 |
|           4 | LCD                  |   5 |   6 |
|           5 | PLASMA               |   7 |   8 |
|           6 | PORTABLE ELECTRONICS |  10 |  19 |
|           7 | MP3 PLAYERS          |  11 |  12 |
|           8 | CD PLAYERS           |  13 |  14 |
|           9 | MÓVILES              |  15 |  16 |
+-------------+----------------------+-----+-----+
```

Utilizamos valores diferentes a ‘left’ y ‘right’ debido a que estas palabras están reservadas en MySQL. Para determinar los valores del margen iniciaremos la numeración el margen mayor izquierdo e iremos asignando índices a cada margen que encontremos avanzando hacia la derecha.

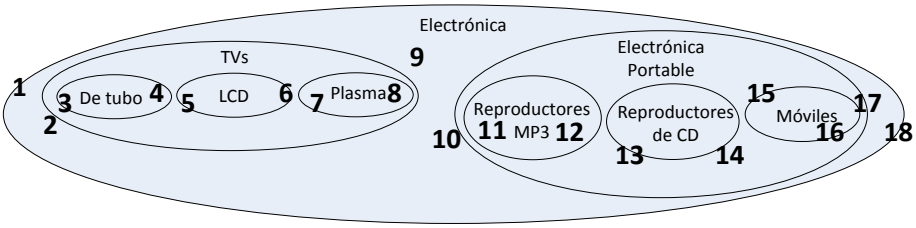


Figura 45 – Representación anidada del modelo del estudio con los índices de cota incluidos.

Este diseño se puede presentar según el modelo clásico de jerarquía como:

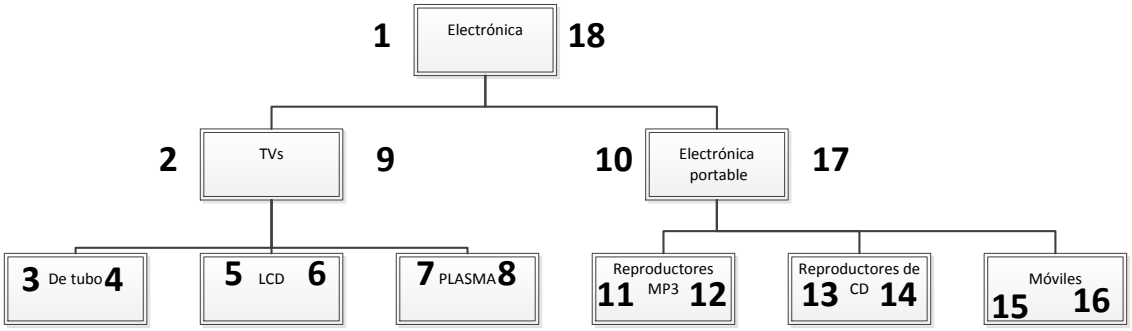


Figura 46 – Adhesión de los índices al modelo del estudio.

Al manipular el árbol trabajamos de izquierda a derecha, un nivel por cada turno, descendiendo por cada elemento nodo hijo antes de asignar el valor correspondiente al valor derecho y cambiar de rama a la derecha siguiente, Este enfoque se denomina 'algoritmo transversal de preordinación modificada del árbol'.

VII.3.1. Recuperar el árbol completo

Podemos recuperar el árbol completo mediante la utilización de un self-join. Los enlaces padre con valor 'lft' de un nodo aparecerán contenidos entre los valores lft y rgt de su padre.

```
SELECT node.name
FROM nested_category AS node,
        nested_category AS parent
WHERE node.lft BETWEEN parent.lft AND parent.rgt
        AND parent.name = 'ELECTRONICS'
ORDER BY node.lft;

+----------------------+
| name                 |
+----------------------+
| ELECTRONICS          |
| TELEVISIONS          |
| TUBE                 |
| LCD                  |
| PLASMA               |
| PORTABLE ELECTRONICS |
| MP3 PLAYERS          |
| CD PLAYERS           |
| MÓVILES         |
+----------------------+
```

A diferencia del ejemplo anterior, esta consulta nos permite obtener el resultado esperado independientemente del nivel de profundidad (anidación) en el árbol. No debemos preocuparnos sobre el valor rgt del nodo debido a que nuestra cláusula siempre estará contenida en el padre al igual que el valor lft.

VII.3.2. Obtener los elementos finales (hojas)

La obtención de los nodos finales del modelo siguiendo la estructura de anidación aún es más simple que en el método de adyacencia. Si nos fijamos en la tabla encontramos el patrón que los elementos finales siempre tienen valores consecutivos entre el campo lft y rgt. Por ese motivo, para obtener los elementos finales será tan fácil como consultar en la tabla la devolución de cualquier nodo que cumpla la regla: rgt = lft +1.

```
SELECT name
FROM nested_category
WHERE rgt = lft + 1;

+--------------+
| name         |
+--------------+
| TUBE         |
| LCD          |
| PLASMA       |
| MP3 PLAYER   |
| CD PLAYERS   |
| MÓVIL        |
+--------------+
```

VII.3.3. Obtener una rama concreta

Con este método conseguimos obtener una rama específica sin un número elevado de self-joins

```
SELECT parent.name
FROM nested_category AS node,
        nested_category AS parent
WHERE node.lft BETWEEN parent.lft AND parent.rgt
        AND node.name = 'MP3 PLAYERS'
ORDER BY node.lft;

+----------------------+
| name                 |
+----------------------+
| ELECTRONICS          |
| PORTABLE ELECTRONICS |
| MP3 PLAYERS          |
+----------------------+
```

VII.3.4. Obtener la profundidad de cada nodo en la jerarquía

Ahora que ya hemos visto cómo obtener todo el árbol de un nodo concreto, en este caso deseamos conocer la profundidad de anidación en la que se encuentra. Esto se puede hacer mediante la combinación de la función COUNT junto con a de GROUP BY.

```
SELECT node.name, (COUNT(parent.name) - 1) AS depth
FROM nested_category AS node,
        nested_category AS parent
WHERE node.lft BETWEEN parent.lft AND parent.rgt
GROUP BY node.name
ORDER BY node.lft;

+----------------------+-------+
| name                 | depth |
+----------------------+-------+
| ELECTRONICS          |     0 |
| TELEVISIONS          |     1 |
| TUBE                 |     2 |
| LCD                  |     2 |
| PLASMA               |     2 |
| PORTABLE ELECTRONICS |     1 |
| MP3 PLAYERS          |     2 |
| CD PLAYERS           |     2 |
| 2 WAY RADIOS         |     2 |
+----------------------+-------+
```

VII.3.5. Obtener la profundidad de una rama concreta

Cuando necesitamos el nivel de profundidad de una sub-rama no podemos limitar la búsqueda a un SELF-JOIN ya que corromperá los resultados. En logar de esto, realizamos un tercer SELF-JOIN junto con una sub-consulta que determine la profundidad que será el punto de partida para el sub-árbol.

```
SELECT node.name, (COUNT(parent.name) - (sub_tree.depth + 1)) AS depth
FROM nested_category AS node,
        nested_category AS parent,
        nested_category AS sub_parent,
        (
                SELECT node.name, (COUNT(parent.name) - 1) AS depth
                FROM nested_category AS node,
                nested_category AS parent
                WHERE node.lft BETWEEN parent.lft AND parent.rgt
                AND node.name = 'PORTABLE ELECTRONICS'
                GROUP BY node.name
                ORDER BY node.lft
        )AS sub_tree
WHERE node.lft BETWEEN parent.lft AND parent.rgt
        AND node.lft BETWEEN sub_parent.lft AND sub_parent.rgt
        AND sub_parent.name = sub_tree.name
GROUP BY node.name
ORDER BY node.lft;

+----------------------+-------+
| name                 | depth |
+----------------------+-------+
| PORTABLE ELECTRONICS |     0 |
| MP3 PLAYERS          |     1 |
| CD PLAYERS           |     1 |
| MÓVILES              |     1 |
+----------------------+-------+
```

Esta función puede ser utilizada para cualquier nodo, incluyendo también el elemento raíz. Los valores de profundidad son relativos al nodo consultado.

VII.3.6. Obtener los subordinados inmediatos de un nodo

En caso de querer exclusivamente obtener los elementos subordinados inmediatos deberemos hacer uso de la directriz HAVING de MySQL a la consulta anterior.

```
SELECT node.name, (COUNT(parent.name) - (sub_tree.depth + 1)) AS depth
FROM nested_category AS node,
        nested_category AS parent,
        nested_category AS sub_parent,
        (
                SELECT node.name, (COUNT(parent.name) - 1) AS depth
                FROM nested_category AS node,
                        nested_category AS parent
                WHERE node.lft BETWEEN parent.lft AND parent.rgt
                        AND node.name = 'PORTABLE ELECTRONICS'
                GROUP BY node.name
                ORDER BY node.lft
        )AS sub_tree
WHERE node.lft BETWEEN parent.lft AND parent.rgt
        AND node.lft BETWEEN sub_parent.lft AND sub_parent.rgt
        AND sub_parent.name = sub_tree.name
GROUP BY node.name
HAVING depth <= 1
ORDER BY node.lft;

+----------------------+-------+
| name                 | depth |
+----------------------+-------+
| PORTABLE ELECTRONICS |     0 |
| MP3 PLAYERS          |     1 |
| CD PLAYERS           |     1 |
| MÓVILES              |     1 |
+----------------------+-------+
```

En caso de no desear obtener el elemento raíz deberíamos cambiar la cláusula 'HAVING depth <= 1' por 'HAVING depth = 1'.

VII.3.7. Añadir nodos

Una vez aprendida la fórmula de obtención de información, resulta igual de interesante la manipulación y actualización de la jerarquía al realizan nuevas inclusiones. Revisemos el modelo previo:

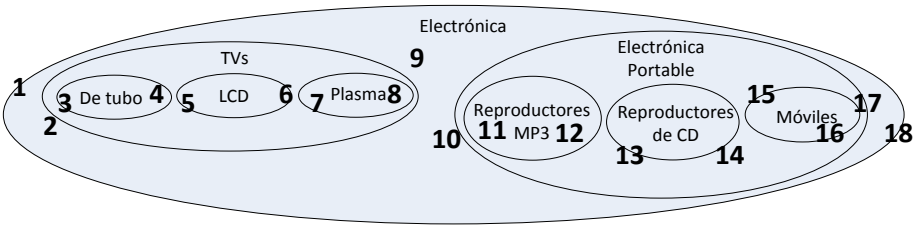


Figura 47 – Representación anidada del modelo con los índices de cota incluidos.

Imaginemos que deseamos introducir un nuevo nodo entre Televisiones y Electrónica portable. Este nuevo nodo debería tener por campos lft y rgt los valores 10 y 11 respectivamente, y después, todos los nodos situados a su derecha, deberían incrementar el valor de estos mismos campos en dos unidades. Una vez modificados, introducir el nuevo nodo. Hay que precisar muy bien la versión MySQL que disponemos. Asumimos que generalmente nos encontramos con la versión 5.0, con la que no hay problema alguno(en caso de utilizar una versión 4.1 o posterior deberemos bloquear la tabla durante el procedimiento de adhesión mediante LOCK TABLE).

```
LOCK TABLE nested_category WRITE;

SELECT @myRight := rgt FROM nested_category
WHERE name = 'TELEVISIONS';

UPDATE nested_category SET rgt = rgt + 2 WHERE rgt > @myRight;
UPDATE nested_category SET lft = lft + 2 WHERE lft > @myRight;

INSERT INTO nested_category(name, lft, rgt) VALUES('GAME CONSOLES', @myRight + 1,
@myRight + 2);

UNLOCK TABLES;
```

Se puede comprobar el lugar dónde anidar el nodo mediante la consulta al árbol siguiente:

```
SELECT CONCAT( REPEAT( ' ', (COUNT(parent.name) - 1) ), node.name) AS name
FROM nested_category AS node,
        nested_category AS parent
WHERE node.lft BETWEEN parent.lft AND parent.rgt
GROUP BY node.name
ORDER BY node.lft;

+-----------------------+
| name                  |
+-----------------------+
| ELECTRONICS           |
|  TELEVISIONS          |
|   TUBE                |
|   LCD                 |
|   PLASMA              |
|  GAME CONSOLES        |
|  PORTABLE ELECTRONICS |
|   MP3 PLAYERS         |
|   CD PLAYERS          |
|   MÓVIL               |
+-----------------------+
```

Si por el contrario queremos añadir un nodo como hijo de otro nodo que no tiene hijos anidados, debemos modificar el procedimiento ligeramente. Añadimos un nodo 3G anidado en Móviles.

```
LOCK TABLE nested_category WRITE;

SELECT @myLeft := lft FROM nested_category

WHERE name = 'MÓVILES';

UPDATE nested_category SET rgt = rgt + 2 WHERE rgt > @myLeft;
UPDATE nested_category SET lft = lft + 2 WHERE lft > @myLeft;

INSERT INTO nested_category(name, lft, rgt) VALUES('3G', @myLeft + 1, @myLeft + 2);

UNLOCK TABLES;
```

En este ejemplo, todo aquel valor superior al que introducimos como rgt. Después introducimos el nuevo nodo a la derecha del valor actualizado y se puede observar como el valor está correctamente anidado.

```
SELECT CONCAT( REPEAT( ' ', (COUNT(parent.name) - 1) ), node.name) AS name
FROM nested_category AS node,
        nested_category AS parent
WHERE node.lft BETWEEN parent.lft AND parent.rgt
GROUP BY node.name
ORDER BY node.lft;

+-----------------------+
| name                  |
+-----------------------+
```

```
| ELECTRONICS           |
|  TELEVISIONS          |
|   TUBE                |
|   LCD                 |
|   PLASMA              |
|  GAME CONSOLES        |
|  PORTABLE ELECTRONICS |
|   MP3 PLAYERS         |
|   CD PLAYERS          |
|   MÓVILES             |
|    3G                 |
+-----------------------+
```

VII.3.8. Eliminar nodos

La última tarea básica para la manipulación de datos en BBDD según el modelo de anidación corresponde a la eliminación de estos. La rutina de acción de eliminación de un nodo depende de la posición del nodo en la jerarquía. Eliminar elementos finales es más simple que nodos con hijos ya que no hay que considerar la orfandad de los hijos.

Para la eliminación de un elemento final de la jerarquía (sin hijos) realizamos el proceso inverso al de adición.

```
LOCK TABLE nested_category WRITE;

SELECT @myLeft := lft, @myRight := rgt, @myWidth := rgt - lft + 1
FROM nested_category
WHERE name = 'GAME CONSOLES';

DELETE FROM nested_category WHERE lft BETWEEN @myLeft AND @myRight;

UPDATE nested_category SET rgt = rgt - @myWidth WHERE rgt > @myRight;
UPDATE nested_category SET lft = lft - @myWidth WHERE lft > @myRight;

UNLOCK TABLES;
```

De esta forma obtenemos como resultado:

```
SELECT CONCAT( REPEAT( ' ', (COUNT(parent.name) - 1) ), node.name) AS name
FROM nested_category AS node,
        nested_category AS parent
WHERE node.lft BETWEEN parent.lft AND parent.rgt
GROUP BY node.name
ORDER BY node.lft;

+-----------------------+
| name                  |
+-----------------------+
| ELECTRONICS           |
|  TELEVISIONS          |
|   TUBE                |
|   LCD                 |
|   PLASMA              |
|  PORTABLE ELECTRONICS |
|   MP3 PLAYERS         |
|    FLASH              |
|   CD PLAYERS          |
|   MÓVILES             |
|    3G                 |
+-----------------------+
```

Esta misma rutina sirve tanto para la eliminación de un nodo como para la de toda su jerarquía descendiente.

```
LOCK TABLE nested_category WRITE;

SELECT @myLeft := lft, @myRight := rgt, @myWidth := rgt - lft + 1
FROM nested_category
WHERE name = 'MÓVILES';

DELETE FROM nested_category WHERE lft BETWEEN @myLeft AND @myRight;

UPDATE nested_category SET rgt = rgt - @myWidth WHERE rgt > @myRight;
UPDATE nested_category SET lft = lft - @myWidth WHERE lft > @myRight;

UNLOCK TABLES;
```

Que obtenemos como resultado:

```
SELECT CONCAT( REPEAT( ' ', (COUNT(parent.name) - 1) ), node.name) AS name
FROM nested_category AS node,
        nested_category AS parent
WHERE node.lft BETWEEN parent.lft AND parent.rgt
GROUP BY node.name
ORDER BY node.lft;

+-----------------------+
| name                  |
+-----------------------+
| ELECTRONICS           |
|  TELEVISIONS          |
|   TUBE                |
|   LCD                 |
|   PLASMA              |
|  PORTABLE ELECTRONICS |
|   MP3 PLAYERS         |
|   CD PLAYERS          |
+-----------------------+
```

La otra opción que debemos contemplar es la de la eliminación del nodo en cuestión, pero no la de todos sus hijos. En este caso, los nodos secundarios deben moverse hasta el nivel del padre eliminado.

```
LOCK TABLE nested_category WRITE;

SELECT @myLeft := lft, @myRight := rgt, @myWidth := rgt - lft + 1
FROM nested_category
WHERE name = 'PORTABLE ELECTRONICS';

DELETE FROM nested_category WHERE lft = @myLeft;

UPDATE nested_category SET rgt = rgt - 1, lft = lft - 1 WHERE lft BETWEEN @myLeft AND
@myRight;
UPDATE nested_category SET rgt = rgt - 2 WHERE rgt > @myRight;
UPDATE nested_category SET lft = lft - 2 WHERE lft > @myRight;

UNLOCK TABLES;
```

Para este caso hemos restado dos al valor de todos los elementos superiores en valor al del valor rgt de nodo en cuestión. Una vez más, confirmamos que los resultados son los esperados:

```
SELECT CONCAT( REPEAT( ' ', (COUNT(parent.name) - 1) ), node.name) AS name
FROM nested_category AS node,
        nested_category AS parent
WHERE node.lft BETWEEN parent.lft AND parent.rgt
GROUP BY node.name
ORDER BY node.lft;

+---------------+
| name          |
+---------------+
| ELECTRONICS   |
|  TELEVISIONS  |
```

```
|   TUBE         |
|   LCD          |
|   PLASMA       |
|  CD PLAYERS    |
|  MÓVILES       |
|   3G           |
+----------------+
```

VII.4. Conclusiones sobre los modelos

El almacenamiento de datos jerárquicos en bases de datos ha migrado para su optimización, a lo largo de los años, al modelo anidado. Si bien su dificultad de comprensión inicial al ser humano es el principal inconveniente, una vez acostumbrado todo el resto no son más que ventajas. Por este motivo, elegimos este modelo para el almacenamiento de la jerarquía de los nodos una vez encontradas las limitaciones al iniciar primero ésta en el primer modelo.

Existe una gran cantidad de información adicional sobre el almacenamiento de jerarquías en bases de datos tanto en libros como en Internet. En nuestra opinión, la que resulta más clarividente es la contenida en el libro titulado "*Joe Celko's Trees and hierarchies in SQL for smarties*"[34], escrito por el respetado autor en el campo del SLQ avanzado Joe Cleko que se acredita a menudo con el modelo de conjuntos anidados y resulta el autor más prolífico en el tema. En este libro se detallan otros muchos temas más avanzados que no pudimos cubrir en esta sección y proporciona métodos adicionales para la gestión de datos jerárquicos además de los modelos de lista de adyacencia y conjuntos anidados.

ANEXO VIII: FUNCIONAMIENTO DE LAS TABLAS ROUND ROBIN EN MYSQL

VIII.1. Introducción

Una planificación round-robin es un método para seleccionar todos los elementos en un grupo de manera equitativa y en un orden racional, normalmente comenzando por el primer elemento de la lista hasta llegar al último y empezando de nuevo desde el primer elemento.

Una base de datos con planificación round-robin almacena generalmente series temporales de datos como por ejemplo temperatura, carga de CPU, ancho de banda, etc. Los datos se almacenan de manera que la estructura permanece constante evitando así tareas de limpieza y mantenimiento y reduciendo complejidad.

La tecnología MySQL no ofrece ninguna solución de configuración de este modelo de motor de almacenamiento. Existe muy poca información y una vaga documentación sobre esta contienda. Un reducido número de personas ha estado investigando y debatiendo la manera de ofrecer esta posibilidad y han creado algún prototipo de solución.

En este apartado, utilizaremos uno de estos prototipos como punto de partida y ofreceremos una solución más compleja que permita la estructura del modelo de datos que se pretende mostrar.

VIII.2. Implementación del prototipo

Para este apartado asumiremos que partimos de una tabla que deseamos convertir en RRDB.

```
CREATE TABLE statistic (  attribute_key  INT UNSIGNED  NOT NULL DEFAULT '0'  , sta
rt_utime  INT UNSIGNED  NOT NULL DEFAULT '0'  , end_utime  INT UNSIGNED
  DEFAULT NULL  , logging_interval  INT UNSIGNED  NOT NULL DEFAULT '0'  , value
   BIGINT UNSIGNED NOT NULL DEFAULT '0'  , PRIMARY KEY (attribute_key, start_utime
)  , KEY start_time (start_utime) ) ;
```

El siguiente apartado es añadir una columna ‘rrd’ que se encargue del comportamiento RRD. Además, limitaremos el máximo del almacenamiento de la tabla evitando posibles huecos, aumentar la velocidad y permitir inserciones concurrentes. Supongamos que queremos almacenar 25.000 filas en esta tabla.

```
CREATE TABLE statistic_rrd (  rrd_key  INT UNSIGNED  NOT NULL AUTO_INCREM
ENT PRIMARY KEY  , attribute_key  INT UNSIGNED  NOT NULL DEFAULT '0'  , start_uti
me  INT UNSIGNED  NOT NULL DEFAULT '0'  , end_utime  INT UNSIGNED
DEFAULT NULL  , logging_interval  INT UNSIGNED  NOT NULL DEFAULT '0'  , value
```

```
  BIGINT UNSIGNED NOT NULL DEFAULT '0'   , UNIQUE KEY (attribute_key, start_utime)   , K
EY start_time (start_utime) ) ROW_FORMAT = FIXED , MAX_ROWS = 25000000 ;
```

A continuación, añadiremos una tabla donde almacenaremos el 'rrd_key' y la inicializaremos.

```
CREATE TABLE statistic_rrd_key (      rrd_key BIGINT UNSIGNED NOT NULL AUTO_INCREM
ENT PRIMARY KEY ) ; INSERT INTO statistic_rrd_key VALUES (0);
```

VIII.2.1. La lógica RRD

Para simular la lógica imprescindible en un modelo RRD en MySQL hay que utilizar disparadores (triggers) por cada inserción. En este caso y teniendo en cuenta el número máximo de columnas, el disparador tendría la siguiente estructura:

```
DROP TRIGGER IF EXISTS statistic_rrd_ins;
DELIMITER $$
CREATE TRIGGER  statistic_rrd_ins BEFORE INSERT ON statistic_rrd FOR EACH ROW
BEGIN
  SET @rrd_key = 0;
  SET @rows = 25000000;
  -- PK is NULL
  IF NEW.rrd_key = 0 THEN
    SELECT rrd_key + 1 FROM statistic_rrd_key
INTO @rrd_key;     SET NEW.rrd_key = @rrd_key;
  END IF;
  IF (NEW.rrd_key % @rows) THEN
    SET NEW.rrd_key = NEW.rrd_key % @rows;
 ELSE
    SET NEW.rrd_key = @rows;
  END IF;
  UPDATE statistic_rrd_key SET rrd_key = NEW.rrd_key;
END;
$$ DELIMITER ;
```

VIII.2.2. Pruebas con el modelo

A continuación, se reemplazan todas las sentencias INSERT por REPLACE y adoptamos todas las querys UPDATE y DELETE. Con este último cambio debería estar preparado el modelo para el funcionamiento deseado.

```
REPLACE INTO statistic_rrd (attribute_key, start_utime, end_utime, logging_interval, value)  V
ALUES   (ROUND(RAND()*100), UNIX_TIMESTAMP(NOW()), NULL, 100, 123456789);

SELECT * FROM statistic_rrd;
SELECT * FROM statistic_rrd_key;
```

VIII.2.3. Algunas medidas de la prueba

La tabla fue volcada según la siguiente lógica:

```
REPLACE INTO statistic_rrd     (attribute_key, start_utime, end_utime, logging_interval, value)
VALUES     (ROUND(RAND()*100000), UNIX_TIMESTAMP(NOW()), NULL, 100, 123456789) ;
```

Ejecutamos esta instrucción en un AMD Athlon 1350MHz con 1CPU y 1Gb de RAM con IDE de disco de 7200 rpm 53248 veces. La media de tiempo de inserción resultó de 600 inserciones/s, lo que es un resultado bastante aceptable teniendo en cuenta las características del servidor utilizado.

VIII.3. El almacenamiento escalar

Los datos que deseamos almacenar tienen un interés inversamente proporcional al tiempo transcurrido desde su inserción. Esto significa que a más tiempo trascurrido desde su inserción, menor trascendencia informativa percibe el consumidor de esta información. Una posible solución de abordar esta situación es agrupar la información más antigua en bloques mediante su media aritmética.

En nuestro estudio, deseamos almacenar throughput de CPU y de Velocidad de transmisión de enlaces. Para escalar estos datos creamos la siguiente lógica:

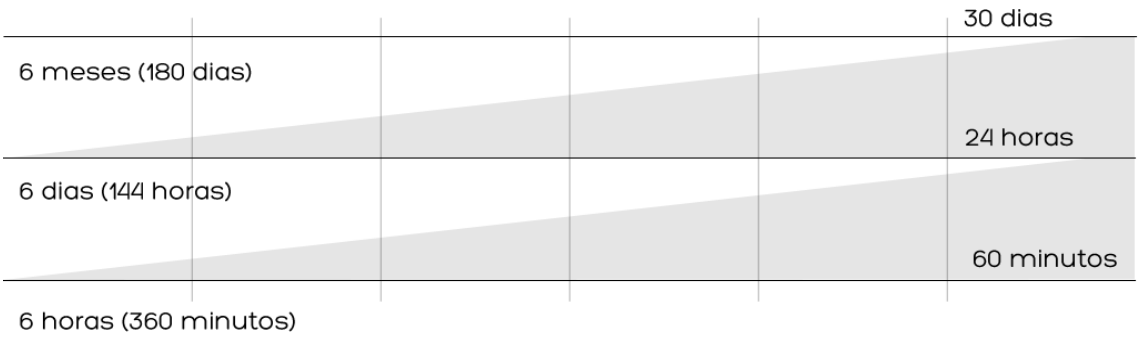


Figura 48 – Escalabilidad de los datos en la RRD del modelo.

VIII.4. La creación de las tablas

Requerimos de la creación del entramado de datos escalables del modelo anterior de manera que necesitaremos tres tablas de estructura Round Robin: minutos, horas y días. Siguiendo la lógica de almacenar para cada tabla lo correspondiente a seis unidades del almacenamiento de la tabla del siguiente nivel nos encontramos con la siguiente configuración de tablas:

	MINUTOS	HORAS	DIAS	MESES
TOTAL tiempo almacenado	360 min	144 horas	180 días	*
correspondencia	6 horas	6 días	6 meses	*
Muestra cada	1 minuto	1 hora	1 día	30 días
Δ(t) en milisegundos	60K	1.44M	86,4M	2.592M

En este caso hemos incluido una cuarta tabla que almacena meses en su totalidad.

VIII.4.1. El almacenamiento de datos minuteros

Para la creación de la tabla minutera siguiendo el prototipo escogido existe la siguiente instrucción:

```
DROP TABLE IF EXISTS RRD_[node_name]_min

CREATE TABLE `RRD_[node_name]_min` (
 `rrd_key` int(10) unsigned NOT NULL AUTO_INCREMENT,
 `attribute_key` int(10) unsigned NOT NULL DEFAULT '0',
 `start_utime` int(10) unsigned NOT NULL DEFAULT '0',
 `logging_interval` int(10) unsigned NOT NULL DEFAULT '0',
 PRIMARY KEY (`rrd_key`),
 UNIQUE KEY `attribute_key` (`attribute_key`,`start_utime`),
 KEY `start_time` (`start_utime`)
) ENGINE=MyISAM AUTO_INCREMENT=361 DEFAULT CHARSET=latin1 MAX_ROWS=360
ROW_FORMAT=FIXED;

ALTER TABLE `RRD_[node_name]_min` ADD `VAL` bigint(20) unsigned NOT NULL
DEFAULT '0';
```

De toda la instrucción debemos destacar la última línea de ejecución en la que se añade a la tabla una columna con nombre ‘VAL’ que corresponde con el almacenamiento del throughput del nodo.

A semejanza de esta instrucción, hay que añadir tantas como enlaces cuyo origen parta de este mismo nodo:

```
ALTER TABLE `RRD_[node_name]_min` ADD `[id_source]_[id_link]` bigint(20) unsigned NOT
NULL DEFAULT '0';
```

A continuación hay que añadir una tabla que actúe de índice de esta tabla:

```
DROP TABLE IF EXISTS `RRD_[node_name]_min_key`;

CREATE TABLE `RRD_[node_name]_min_key` (
  `rrd_key` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`rrd_key`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

LOCK TABLES `RRD_[node_name]_min_key` WRITE;
/*!40000 ALTER TABLE `RRD_[node_name]_min_key` DISABLE KEYS */;

INSERT INTO `RRD_[node_name]_min_key` (`rrd_key`)
VALUES (0);
```

De la misma forma se debe ejecutar las tablas referentes al almacenamiento de horas y días.

VIII.4.2. El almacenamiento de datos horarios

```
DROP TABLE IF EXISTS `RRD_[node_name]_hour`;

CREATE TABLE `RRD_[node_name]_hour` (
  `rrd_key` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `attribute_key` int(10) unsigned NOT NULL DEFAULT '0',
  `start_utime` int(10) unsigned NOT NULL DEFAULT '0',
  `logging_interval` int(10) unsigned NOT NULL DEFAULT '0',
  PRIMARY KEY (`rrd_key`),
  UNIQUE KEY `attribute_key` (`attribute_key`,`start_utime`),
  KEY `start_time` (`start_utime`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 MAX_ROWS=144 ROW_FORMAT=FIXED;

ALTER TABLE `RRD_[node_name]_hour` ADD `VAL` bigint(20) unsigned NOT NULL
DEFAULT '0';
```

Y para cada enlace cuyo nodo origen sea la id del nodo en cuestión:

```
ALTER TABLE `RRD_[node_name]_hour` ADD `[id_source]_[id_link]` bigint(20) unsigned NOT
NULL DEFAULT '0';
```

Y por último:

```
DROP TABLE IF EXISTS `RRD_[node_name]_hour_key`;

CREATE TABLE `RRD_[node_name]_hour_key` (
  `rrd_key` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`rrd_key`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

LOCK TABLES `RRD_[node_name]_hour_key` WRITE;
/*!40000 ALTER TABLE `RRD_[node_name]_hour_key` DISABLE KEYS */;

INSERT INTO `RRD_[node_name]_hour_key` (`rrd_key`)
VALUES
        (0);
```

VIII.4.3. El almacenamiento de datos diarios

```
DROP TABLE IF EXISTS `RRD_[node_name]_day`;

CREATE TABLE `RRD_[node_name]_day` (
  `rrd_key` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `attribute_key` int(10) unsigned NOT NULL DEFAULT '0',
  `start_utime` int(10) unsigned NOT NULL DEFAULT '0',
  `logging_interval` int(10) unsigned NOT NULL DEFAULT '0',
  PRIMARY KEY (`rrd_key`),
  UNIQUE KEY `attribute_key` (`attribute_key`,`start_utime`),
```

```
  KEY `start_time` (`start_utime`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 MAX_ROWS=180 ROW_FORMAT=FIXED;

ALTER TABLE `RRD_[node_name]_day` ADD `VAL` bigint(20) unsigned NOT NULL
DEFAULT '0';
```

Y para cada enlace cuyo nodo origen sea la id del nodo en cuestión:

```
ALTER TABLE `RRD_[node_name]_day` ADD `[id_source]_[id_link]` bigint(20) unsigned NOT
NULL DEFAULT '0';
```

VIII.4.4. El almacenamiento de datos mensuales global

```
DROP TABLE IF EXISTS `RRD_[node_name]_day`;

CREATE TABLE `RRD_[node_name]_day` (
  `rrd_key` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `attribute_key` int(10) unsigned NOT NULL DEFAULT '0',
  `start_utime` int(10) unsigned NOT NULL DEFAULT '0',
  `logging_interval` int(10) unsigned NOT NULL DEFAULT '0',
  PRIMARY KEY (`rrd_key`),
  UNIQUE KEY `attribute_key` (`attribute_key`,`start_utime`),
  KEY `start_time` (`start_utime`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 MAX_ROWS=180 ROW_FORMAT=FIXED;

ALTER TABLE `RRD_[node_name]_day` ADD `VAL` bigint(20) unsigned NOT NULL
DEFAULT '0';
```

Y para cada enlace cuyo nodo origen sea la id del nodo en cuestión:

```
ALTER TABLE `RRD_[node_name]_day` ADD `[id_source]_[id_link]` bigint(20) unsigned NOT
NULL DEFAULT '0';
```

Y por último:

```
DROP TABLE IF EXISTS `RRD_[node_name]_day_key`;

CREATE TABLE `RRD_[node_name]_day_key` (
  `rrd_key` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`rrd_key`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

LOCK TABLES `RRD_[node_name]_day_key` WRITE;
/*!40000 ALTER TABLE `RRD_[node_name]_day_key` DISABLE KEYS */;

INSERT INTO `RRD_[node_name]_day_key` (`rrd_key`)
VALUES
        (0);
```

2.4.4 El almacenamiento de datos mensuales global.

```
DROP TABLE IF EXISTS `RRD_[node_name]_month`;
```

```
CREATE TABLE `RRD_[node_name]_month` (
  `rrd_key` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `attribute_key` int(10) unsigned NOT NULL DEFAULT '0',
  `start_utime` int(10) unsigned NOT NULL DEFAULT '0',
  `logging_interval` int(10) unsigned NOT NULL DEFAULT '0',
  PRIMARY KEY (`rrd_key`),
  UNIQUE KEY `attribute_key` (`attribute_key`,`start_utime`),
  KEY `start_time` (`start_utime`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 ROW_FORMAT=FIXED;

ALTER TABLE `RRD_[node_name]_month` ADD `VAL` bigint(20) unsigned NOT NULL
DEFAULT '0';
```

Y para cada enlace cuyo nodo origen sea la id del nodo en cuestión:

```
ALTER TABLE `RRD_[node_name]_month` ADD `[id_source]_[id_link]` bigint(20) unsigned
NOT NULL DEFAULT '0';
```

Esta última tabla no sigue una estructura RRD si no que almacena la media de todos los valores mensuales. Por esto carece del atributo MAXROWS, que la hace circular.

VIII.5. La creación de los disparadores

Para entender la lógica de las tablas rrd hay que retomar la figura 29 del apartado 4.1 del anexo IV.

VIII.5.1. El disparador de la tabla minutera

```
DELIMITER ;;
 `RRD_ [node_name]_hour_ins` BEFORE INSERT ON `RRD_ [node_name]_hour` FOR EACH
ROW BEGIN
 SET @rrd_ [node_name]_key_hour = 0;
 SET @rows_hour = 144;
 SET @rows_hour_mult = @rows_hour / 6;
 IF NEW.rrd_key = 0 THEN
  SELECT rrd_key + 1 FROM RRD_ [node_name]_hour_key INTO @rrd_
[node_name]_key_hour;
  SET NEW.rrd_key = @rrd_ [node_name]_key_hour;
 END IF;
 IF (NEW.rrd_key % (@rows_hour+1)) THEN
  IF NOT (NEW.rrd_key % @rows_hour_mult) THEN
   SET @avg_atribute_key = UNIX_TIMESTAMP(NOW());
   CREATE TEMPORARY TABLE IF NOT EXISTS temp AS ( SELECT `VAL` ,
[array(`[id_source]_[id_link]`)] FROM RRD_ [node_name]_hour WHERE `attribute_key` <=
(SELECT MAX(`attribute_key`) FROM RRD_ [node_name]_hour) ORDER BY `attribute_key`
DESC LIMIT 24 );
   SELECT AVG(`VAL`) FROM temp INTO @avg_val;

[Array([
   SELECT AVG(`[id_source]_[id_link]`) FROM temp INTO @avg_[id_source]_[id_link];
])]

   DROP TEMPORARY TABLE IF EXISTS temp;
```

```
   REPLACE INTO RRD_ [node_name]_day (attribute_key,start_utime,logging_interval,VAL,
[array(`[id_source]_[id_link]`)])
VALUES(@avg_atribute_key,UNIX_TIMESTAMP(NOW()),86400000,@avg_VAL,
[array(@avg_[id_source]_[id_link])]);
  END IF;
  SET NEW.rrd_key = NEW.rrd_key % (@rows_hour+1);
 ELSE
  SET @avg_atribute_key = UNIX_TIMESTAMP(NOW());
  CREATE TEMPORARY TABLE IF NOT EXISTS temp AS ( SELECT `VAL` ,
[array(`[id_source]_[id_link]`)] FROM RRD_ [node_name]_hour WHERE `attribute_key` <=
(SELECT MAX(`attribute_key`) FROM RRD_ [node_name]_hour) ORDER BY `attribute_key`
DESC LIMIT 24 );

[Array([
   SELECT AVG(`[id_source]_[id_link]`) FROM temp INTO @avg_[id_source]_[id_link];
])]

  DROP TEMPORARY TABLE IF EXISTS temp;
  REPLACE INTO RRD_ [node_name]_day (attribute_key,start_utime,logging_interval,VAL,
[array(`[id_source]_[id_link]`)])
VALUES(@avg_atribute_key,UNIX_TIMESTAMP(NOW()),86400000,@avg_VAL,
[array(@avg_[id_source]_[id_link])]);
  SET NEW.rrd_key = 1;
 END IF;
 UPDATE RRD_ [node_name]_hour_key SET rrd_key = NEW.rrd_key;
END */;;
DELIMITER ;
```

VIII.5.2. El disparador de la tabla horaria

```
DELIMITER ;;
 `RRD_ [node_name]_hour_ins` BEFORE INSERT ON `RRD_ [node_name]_hour` FOR EACH
ROW BEGIN
 SET @rrd_ [node_name]_key_hour = 0;
 SET @rows_hour = 144;
 SET @rows_hour_mult = @rows_hour / 6;
 IF NEW.rrd_key = 0 THEN
  SELECT rrd_key + 1 FROM RRD_ [node_name]_hour_key INTO @rrd_
[node_name]_key_hour;
  SET NEW.rrd_key = @rrd_ [node_name]_key_hour;
 END IF;
 IF (NEW.rrd_key % (@rows_hour+1)) THEN
  IF NOT (NEW.rrd_key % @rows_hour_mult) THEN
   SET @avg_atribute_key = UNIX_TIMESTAMP(NOW());
   CREATE TEMPORARY TABLE IF NOT EXISTS temp AS ( SELECT `VAL` ,
[array(`[id_source]_[id_link]`)] FROM RRD_ [node_name]_hour WHERE `attribute_key` <=
(SELECT MAX(`attribute_key`) FROM RRD_ [node_name]_hour) ORDER BY `attribute_key`
DESC LIMIT 24 );
   SELECT AVG(`VAL`) FROM temp INTO @avg_val;

[Array([
   SELECT AVG(`[id_source]_[id_link]`) FROM temp INTO @avg_[id_source]_[id_link];
])]

   DROP TEMPORARY TABLE IF EXISTS temp;
   REPLACE INTO RRD_ [node_name]_day (attribute_key,start_utime,logging_interval,VAL,
[array(`[id_source]_[id_link]`)])
VALUES(@avg_atribute_key,UNIX_TIMESTAMP(NOW()),86400000,@avg_VAL,
```

```
[array(@avg_[id_source]_[id_link])]);
  END IF;
  SET NEW.rrd_key = NEW.rrd_key % (@rows_hour+1);
 ELSE
  SET @avg_atribute_key = UNIX_TIMESTAMP(NOW());
  CREATE TEMPORARY TABLE IF NOT EXISTS temp AS ( SELECT `VAL` ,
[array(`[id_source]_[id_link]`)] FROM RRD_ [node_name]_hour WHERE `attribute_key` <=
(SELECT MAX(`attribute_key`) FROM RRD_ [node_name]_hour) ORDER BY `attribute_key`
DESC LIMIT 24 );

[Array([
   SELECT AVG(`[id_source]_[id_link]`) FROM temp INTO @avg_[id_source]_[id_link];
])]

  DROP TEMPORARY TABLE IF EXISTS temp;
  REPLACE INTO RRD_ [node_name]_day (attribute_key,start_utime,logging_interval,VAL,
[array(`[id_source]_[id_link]`)])
VALUES(@avg_atribute_key,UNIX_TIMESTAMP(NOW()),86400000,@avg_VAL,
[array(@avg_[id_source]_[id_link])]);
  SET NEW.rrd_key = 1;
 END IF;
 UPDATE RRD_ [node_name]_hour_key SET rrd_key = NEW.rrd_key;
END */;;
DELIMITER ;
```

VIII.5.3. El disparador de la tabla diaria

```
DELIMITER ;;
 `RRD_ [node_name]_day_ins` BEFORE INSERT ON `RRD_ [node_name]_day` FOR EACH
ROW BEGIN
 SET @rrd_ [node_name]_key_day = 0;
 SET @rows_day = 180;
 SET @rows_day_mult = @rows_day / 6;
 IF NEW.rrd_key = 0 THEN
  SELECT rrd_key + 1 FROM RRD_ [node_name]_day_key INTO @rrd_
[node_name]_key_day;
  SET NEW.rrd_key = @rrd_ [node_name]_key_day;
 END IF;
 IF (NEW.rrd_key % (@rows_day+1)) THEN
  IF NOT (NEW.rrd_key % @rows_day_mult) THEN
   SET @avg_atribute_key = UNIX_TIMESTAMP(NOW());
   CREATE TEMPORARY TABLE IF NOT EXISTS temp AS ( SELECT `VAL` ,
[array(`[id_source]_[id_link]`)] FROM RRD_ [node_name]_day WHERE `attribute_key` <=
(SELECT MAX(`attribute_key`) FROM RRD_ [node_name]_day) ORDER BY `attribute_key`
DESC LIMIT 30 );
   SELECT AVG(`VAL`) FROM temp INTO @avg_val;

[Array([
   SELECT AVG(`[id_source]_[id_link]`) FROM temp INTO @avg_[id_source]_[id_link];
])]

   DROP TEMPORARY TABLE IF EXISTS temp;
   REPLACE INTO RRD_ [node_name]_month (attribute_key,start_utime,logging_interval,VAL,
[array(`[id_source]_[id_link]`)])
VALUES(@avg_atribute_key,UNIX_TIMESTAMP(NOW()),1440237098,@avg_VAL,
[array(@avg_[id_source]_[id_link])]);
  END IF;
  SET NEW.rrd_key = NEW.rrd_key % (@rows_day+1);
```

```
 ELSE
  SET @avg_atribute_key = UNIX_TIMESTAMP(NOW());
  CREATE TEMPORARY TABLE IF NOT EXISTS temp AS ( SELECT `VAL` ,
[array(`[id_source]_[id_link]`)] FROM RRD_ [node_name]_day WHERE `attribute_key` <=
(SELECT MAX(`attribute_key`) FROM RRD_ [node_name]_day) ORDER BY `attribute_key`
DESC LIMIT 30 );

[Array([
   SELECT AVG(`[id_source]_[id_link]`) FROM temp INTO @avg_[id_source]_[id_link];
])]

  DROP TEMPORARY TABLE IF EXISTS temp;
  REPLACE INTO RRD_ [node_name]_month (attribute_key,start_utime,logging_interval,VAL,
[array(`[id_source]_[id_link]`)])
VALUES(@avg_atribute_key,UNIX_TIMESTAMP(NOW()),1440237098,@avg_VAL,
[array(@avg_[id_source]_[id_link])]);
  SET NEW.rrd_key = 1;
 END IF;
 UPDATE RRD_ [node_name]_day_key SET rrd_key = NEW.rrd_key;
END */;;
DELIMITER ;
```

ANEXO IX: APROVISIONAMIENTO DE DATOS

La documentación de los datos introducidos en la memoria no fue una labor sencilla. Existe poca documentación pública en la red y la que hay, además, es poco precisa.

Este anexo recoge las distintas fuentes de información que utilizamos para la simulación de los datos.

IX.1. RedIRIS NOVA.

RedIRIS es la NREN española. En su web[42] incluye un mapa sobre su topología.

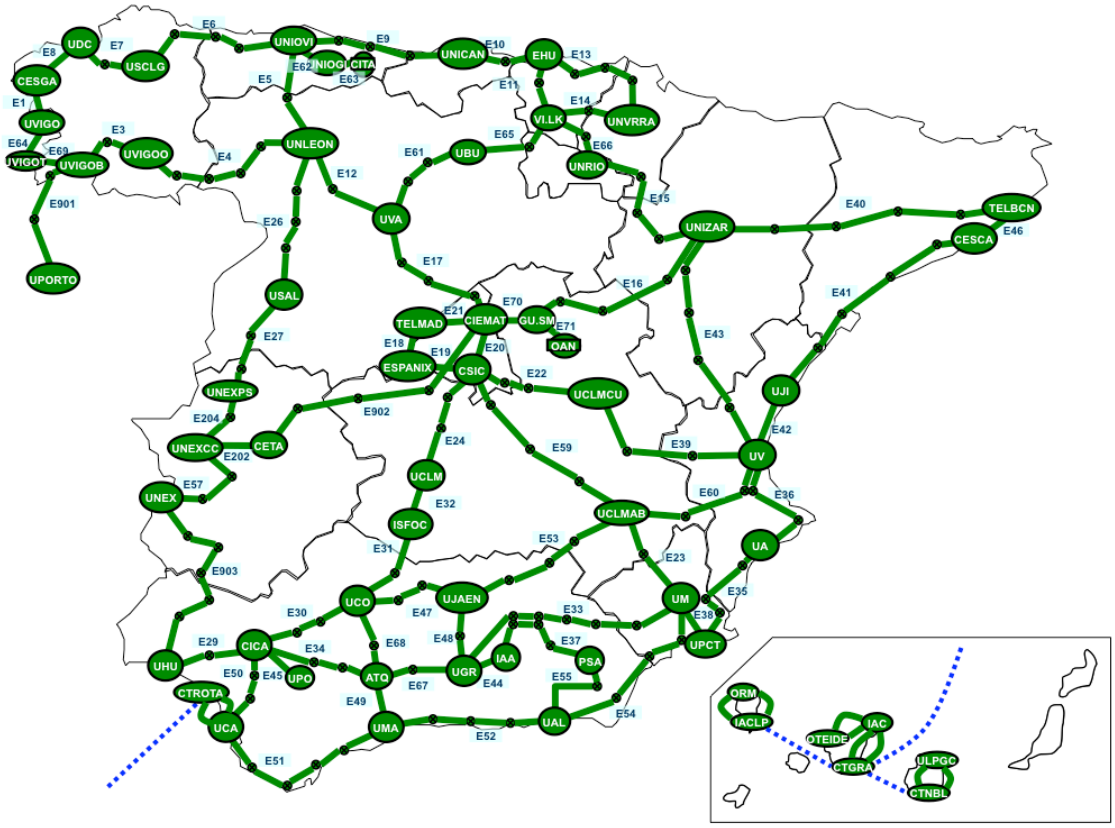


Figura 49 – Mapa de la red RedIRIS-Nova[42].

IX.2. GÉANT

GÉANT es la red pan-europea de interconexión de NRENs. Para la documentación sobre sus distintas redes e interconexiones utilizamos la visualización de la sección privada[43].

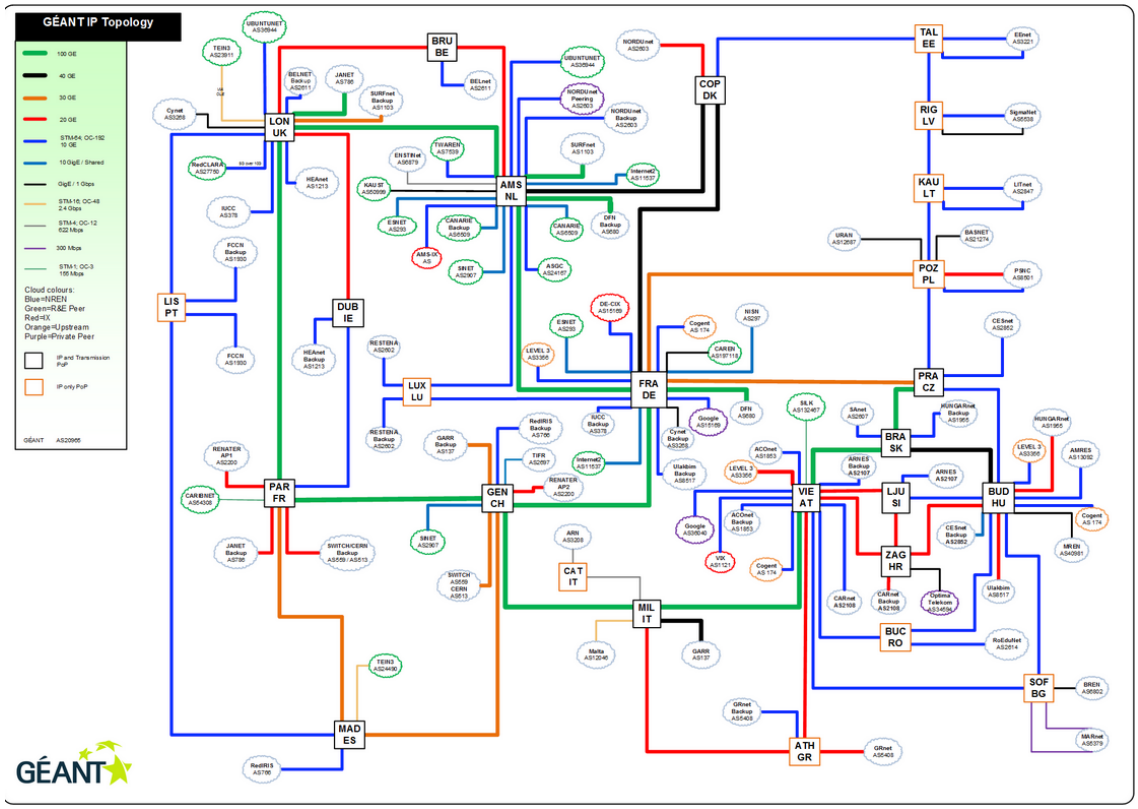


Figura 50 –Topología de la red GÉANT[43].

IX.3. AARNET

AARNet es la NREN de Australia. Utilizamos esta red como garantía soporte de visualizaciones topográficas de gran distancia y a su vez por que tenía todos sus informes públicos y actualizados.

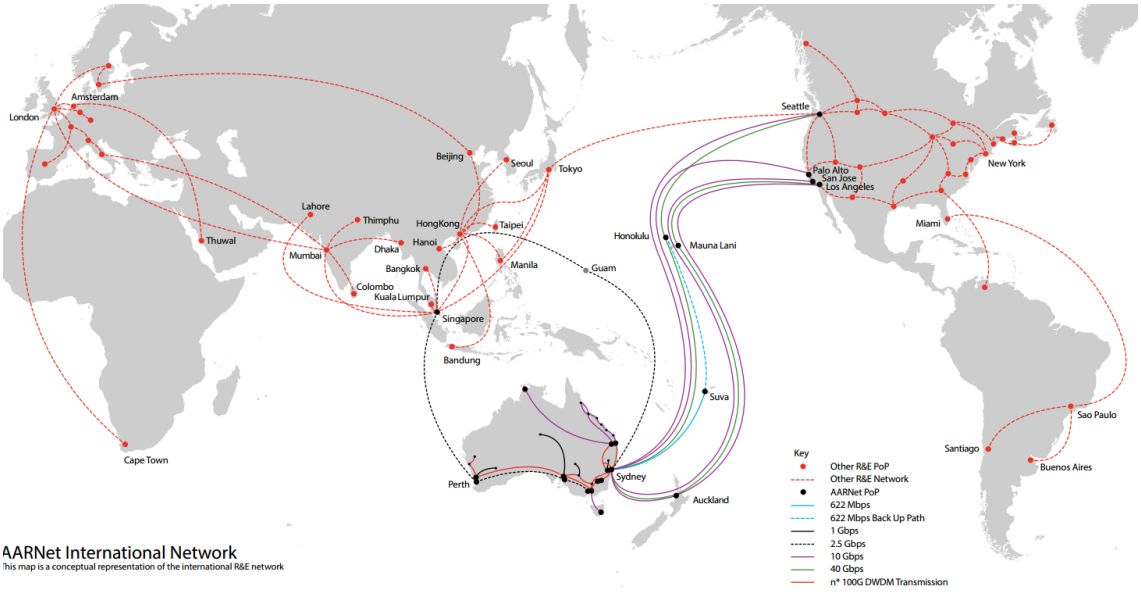


Figura 51 – AARNet International Network[44].

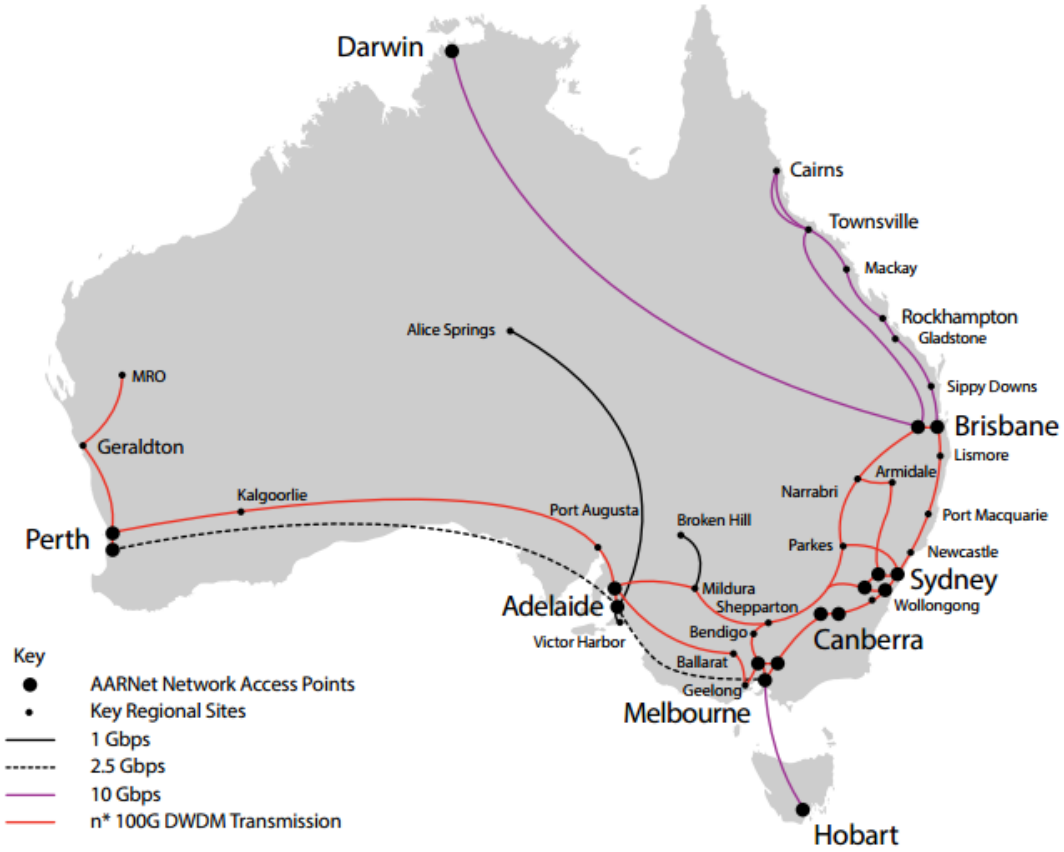


Figura 52 – AARNet National Intercapital Network[45].

IX.4. Topology Zoo

Encontramos un directorio público[46] de topology-zoo que contenía los archivos de estructura de redes NREN mundiales. Introdujimos algunos valores para comprobar la estabilidad de la aplicación a pesar de ser de dudosa fiabilidad debido a su antigüedad (Julio de 2012).

ANEXO X: ANALISIS DE VISUALIZADORES DE DATOS DE RED

X.1. Contexto

Si consideramos esta situación como el mejor panorama para la creación de una opción que alberge cada una de las soluciones por las que apuestan estas herramientas, es preciso estudiarlas y conocer sus principales ventajas e inconvenientes.

Alguna de las principales herramientas que inicialmente aparecieron para la monitorización y análisis de tráfico en la red son: TCPDump, Wireshark, MRTG y NetFlow (NFDump-NFSen), CACTI and Php Network Weathermap.

X.2. Herramientas existentes

Este apartado resume las principales características de los analizadores de datos de red mencionados.

X.2.1. TCPDUMP

Es una herramienta en línea de comandos cuya utilidad es analizar el tráfico que circula por la red. Permite capturar y mostrar en tiempo real los paquetes transmitidos y recibidos de la red a la que el cliente se encuentra conectado.

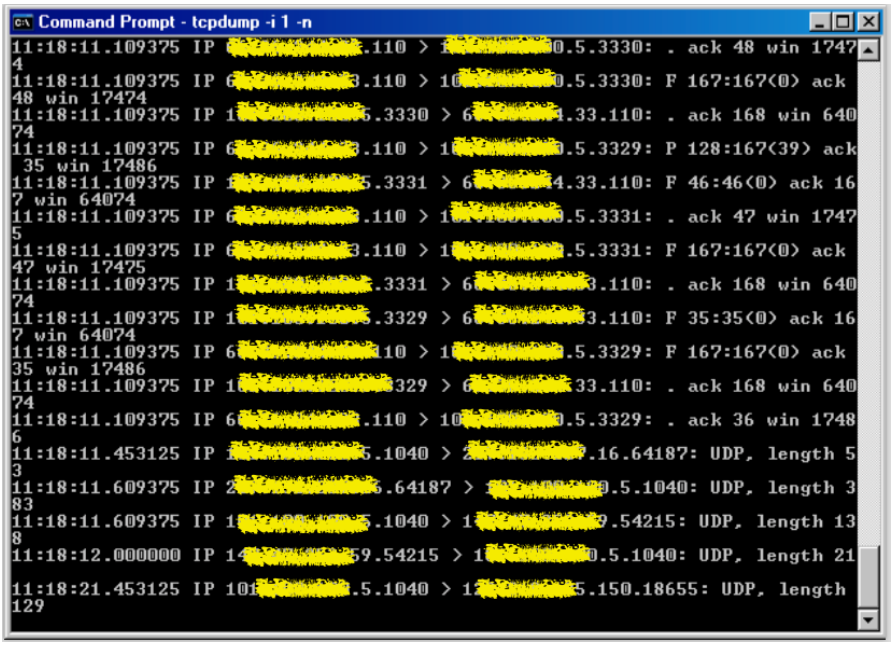


Figura 53 – Interfaz de visualización TCPDump.

VENTAJAS:

- No es código propietario

INCONVENIENTES:

- Exclusivo para monitorizar tráfico local
- La interfaz de comandos es poco visual y en la actualidad se tiende a presentar los resultados de una manera más cómoda y atractiva al usuario.

X.2.2. Wireshark

Es la herramienta analizadora de protocolos de red principal de internet. Permite visualizar lo que ocurre en la red analizada a un nivel microscópico.

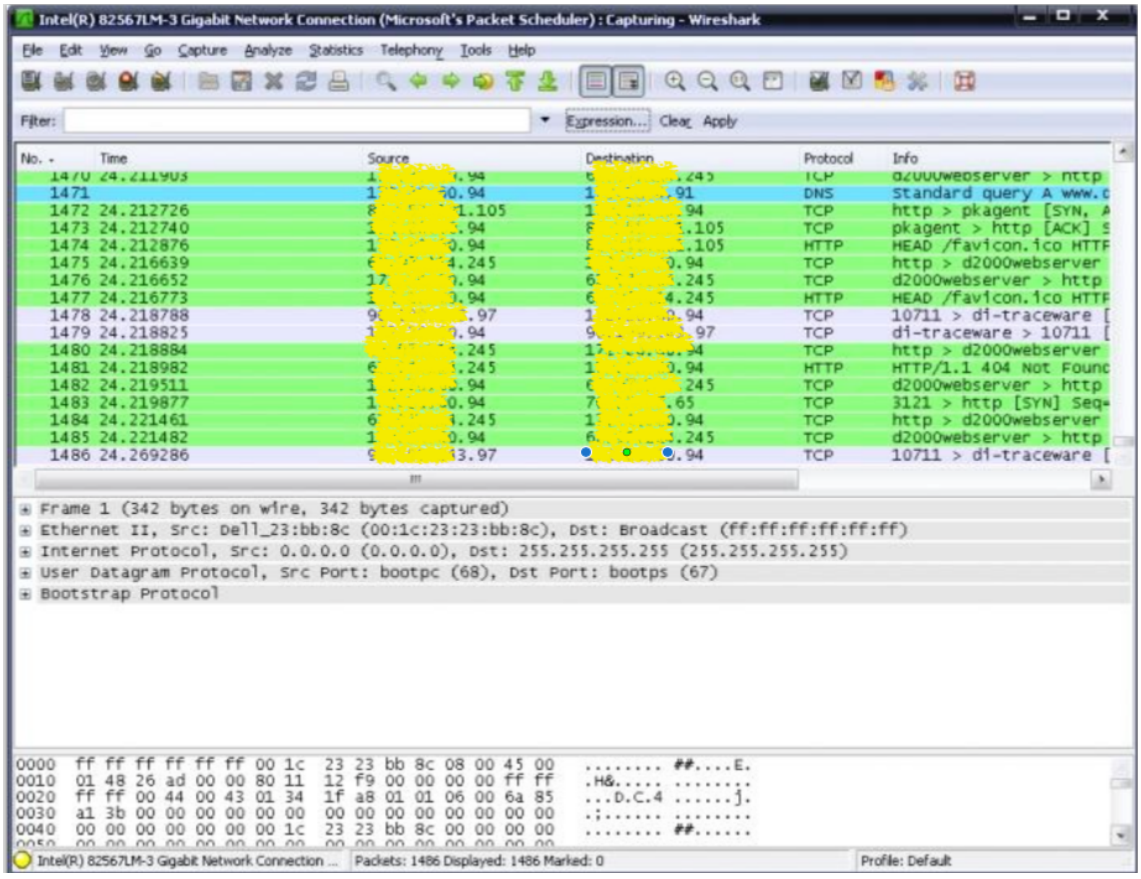


Figura 54 –Interfaz de visualización de Wireshark.

VENTAJAS:

- Primer programa que proporciona una GUI para la monitorización del tráfico.

INCONVENIENTES:

- Los datos obtenidos mediante esta interfaz son de difícil asimilación para el administrador de red invirtiendo un tiempo considerable para su análisis. A pesar de ello, resulta una herramienta todavía muy útil para el análisis específico de determinados comportamientos al presentar los datos a muy bajo nivel pero de una forma muy amigable.

X.2.3. MRTG

Es una herramienta, escrita en C y Perl que se utiliza para supervisar la carga de tráfico de interfaces de red. MRTG genera un informe en formato HTML con gráficas que proveen una representación visual de la evolución del tráfico a lo largo del tiempo.

Para recoger la información del tráfico del dispositivo, la herramienta utiliza el protocolo SNMP (Simple Network Management Protocol). Este protocolo proporciona la información en crudo de la cantidad de bytes que han pasado por ellos distinguiendo entre entrada y salida. Esta cantidad bruta deberá ser tratada adecuadamente para la generación de informes.

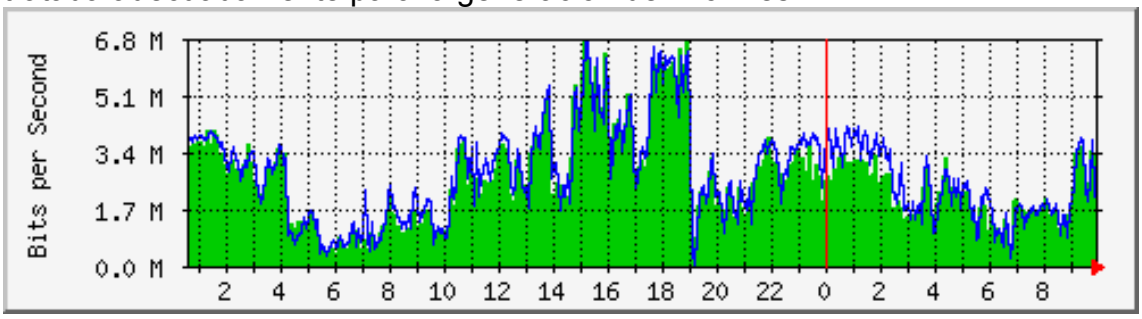


Figura 55 – Interfaz MRTG sobre navegador web.

VENTAJAS

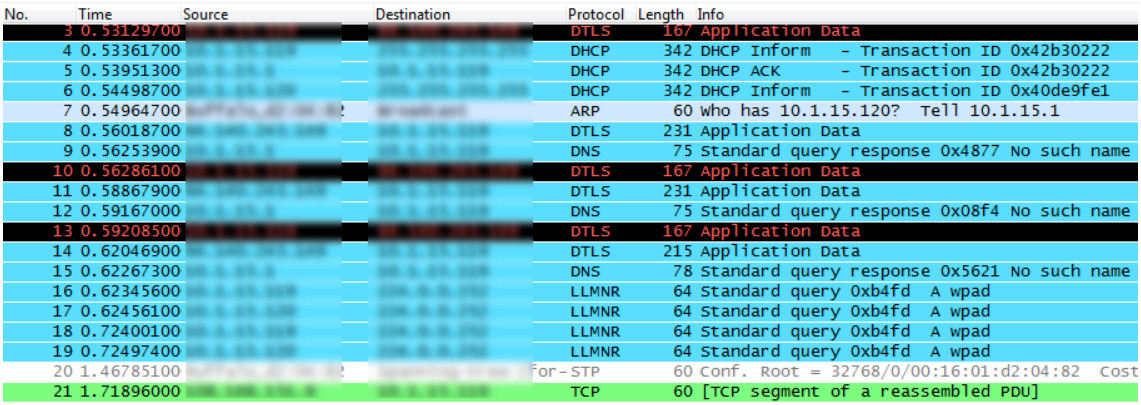
- Al tratarse de una RRDB no se pierde ninguna información histórica del estado del dispositivo.

INCONVENIENTES

- El hecho de estar escalado no permite la vista al detalle de un determinado periodo de tiempo concreto
- Debido a su representación como imagen resulta imposible la interactuación con el usuario y la visualización se resume únicamente a la interpretación de ésta.
- En caso de recibir nueva información, la única forma de representar estos resultados es re-generando la imagen de nuevo con una actualización del directorio de su presentación (URL).

X.2.4. NetFlow y sus visualizadores NFsen, NFdump y PMACCT

Netflow és una técnica de monitorización de flujos de red. También se le da el mismo nombre al protocolo que transporta la información monitorizada (Ver figura 56). Netflow se ha convertido en un estándar de la industria para monitorización de tráfico de red, y actualmente está soportado para varias plataformas además de Cisco, IOS y NXOS y en sistemas operativos como Linux, FreeBSD, NetBSD y OpenBSD.



No.	Time	Source	Destination	Protocol	Length	Info
3	0.53129700	[redacted]	[redacted]	DTLS	167	Application Data
4	0.53361700	[redacted]	[redacted]	DHCP	342	DHCP Inform - Transaction ID 0x42b30222
5	0.53951300	[redacted]	[redacted]	DHCP	342	DHCP ACK - Transaction ID 0x42b30222
6	0.54498700	[redacted]	[redacted]	DHCP	342	DHCP Inform - Transaction ID 0x40de9fe1
7	0.54964700	[redacted]	[redacted]	ARP	60	Who has 10.1.15.120? Tell 10.1.15.1
8	0.56018700	[redacted]	[redacted]	DTLS	231	Application Data
9	0.56253900	[redacted]	[redacted]	DNS	75	Standard query response 0x4877 No such name
10	0.56286100	[redacted]	[redacted]	DTLS	167	Application Data
11	0.58867900	[redacted]	[redacted]	DTLS	231	Application Data
12	0.59167000	[redacted]	[redacted]	DNS	75	Standard query response 0x08f4 No such name
13	0.59208500	[redacted]	[redacted]	DTLS	167	Application Data
14	0.62046900	[redacted]	[redacted]	DTLS	215	Application Data
15	0.62267300	[redacted]	[redacted]	DNS	78	Standard query response 0x5621 No such name
16	0.62345600	[redacted]	[redacted]	LLMNR	64	Standard query 0xb4fd A wpad
17	0.62456100	[redacted]	[redacted]	LLMNR	64	Standard query 0xb4fd A wpad
18	0.72400100	[redacted]	[redacted]	LLMNR	64	Standard query 0xb4fd A wpad
19	0.72497400	[redacted]	[redacted]	LLMNR	64	Standard query 0xb4fd A wpad
20	1.46785100	[redacted]	[redacted]	For-STP	60	Conf. Root = 32768/0/00:16:01:d2:04:82 Cost
21	1.71896000	[redacted]	[redacted]	TCP	60	[TCP segment of a reassembled PDU]

Figura 56 – Captura de paquete con información de protocolo Netflow.

Existen muchos visualizadores de este protocolo de distintas empresas como solarwinds, manageengine, paessler, ireo, todas ellas de pago, y otras muchas gratuitas y de código abierto como son NTop, Flow-tools, FlowScan, EHNT, BPFT y la más popular y extendida NFDUMP/NFSEN (Ver figura 57).

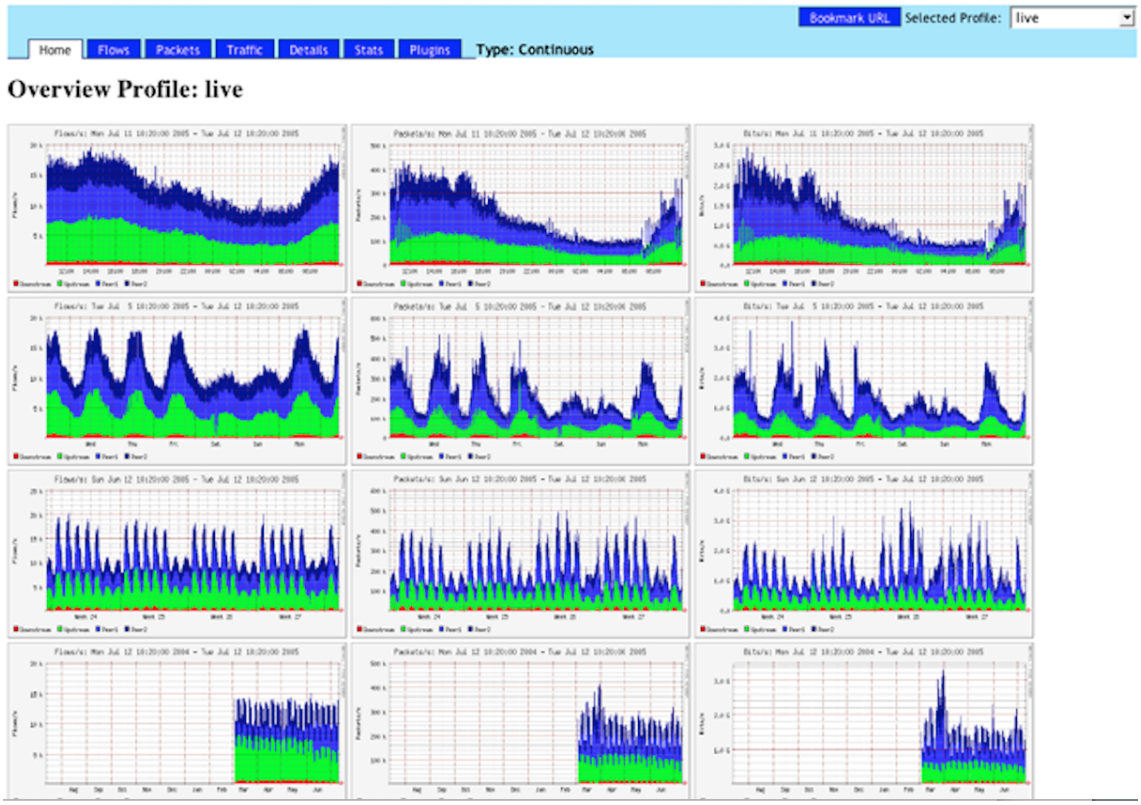


Figura 57 – NfSen General Overview Page.

Centrándonos en esta última como herramienta a estudio obtenemos también una serie de ventajas e inconvenientes:

VENTAJAS

- Navegar con facilidad por los datos de NetFlow
- Procesar los datos dentro de una ventana de tiempo
- Crear archivos históricos y perfiles continuos

INCONVENIENTES

- El panel de control tiene un uso muy complejo debido a la gran cantidad de información de acceso disponible y a veces puede ser incómodo.

X.2.5. Cacti

Cacti es una aplicación gráfica de medición y análisis de código abierto. Su primera versión fue publicada en septiembre del 2001 y proporcionó una completa solución RRDtool en cuanto a front-end sobre plataforma web. Cacti almacena toda la información necesaria para acumular estos datos y generar visualizaciones en una base de datos MySQL. Todas sus funcionalidades son completamente configurables desde una interfaz web. (Ver figuras 58 y 59)

Para el acopio de datos Cacti utiliza scripts y comandos externos así como las 3 versiones SNMP[3].

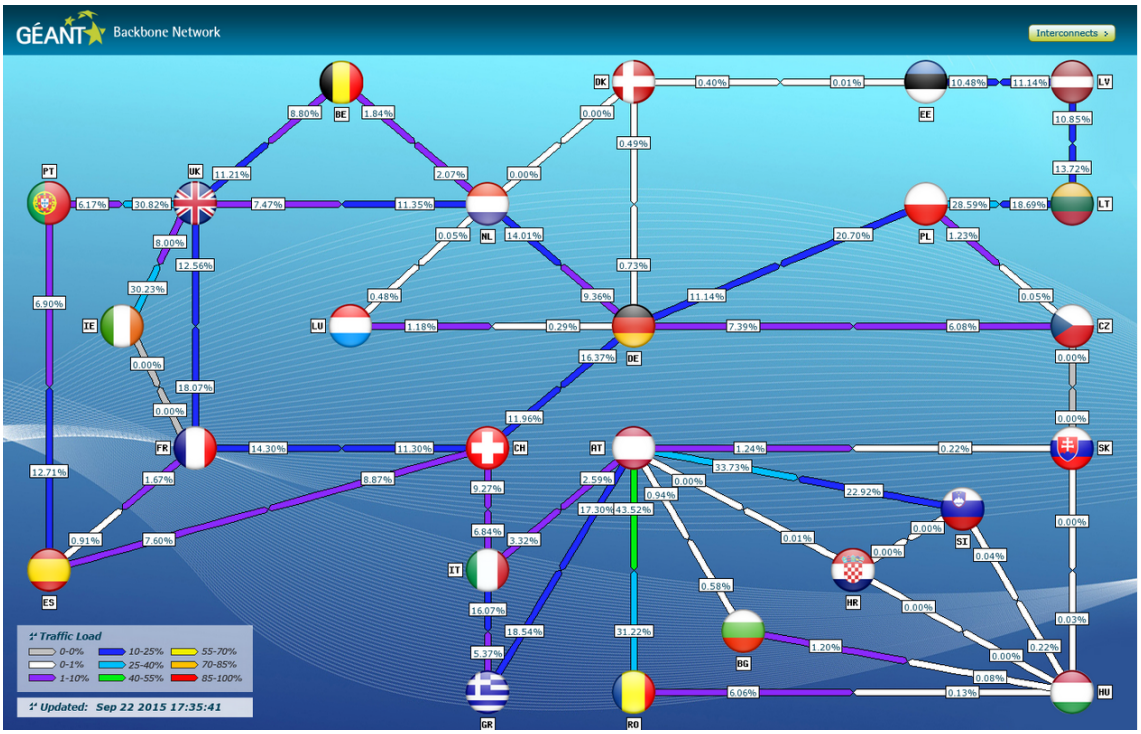


Figura 58 – GÉANT Cacti Weathermap.

[3] SNMP: protocolo de la capa de aplicación que facilita el intercambio de información de administración entre dispositivos de red permitiendo a los administradores supervisar el funcionamiento, planificar su evolución y buscar y resolver sus problemas

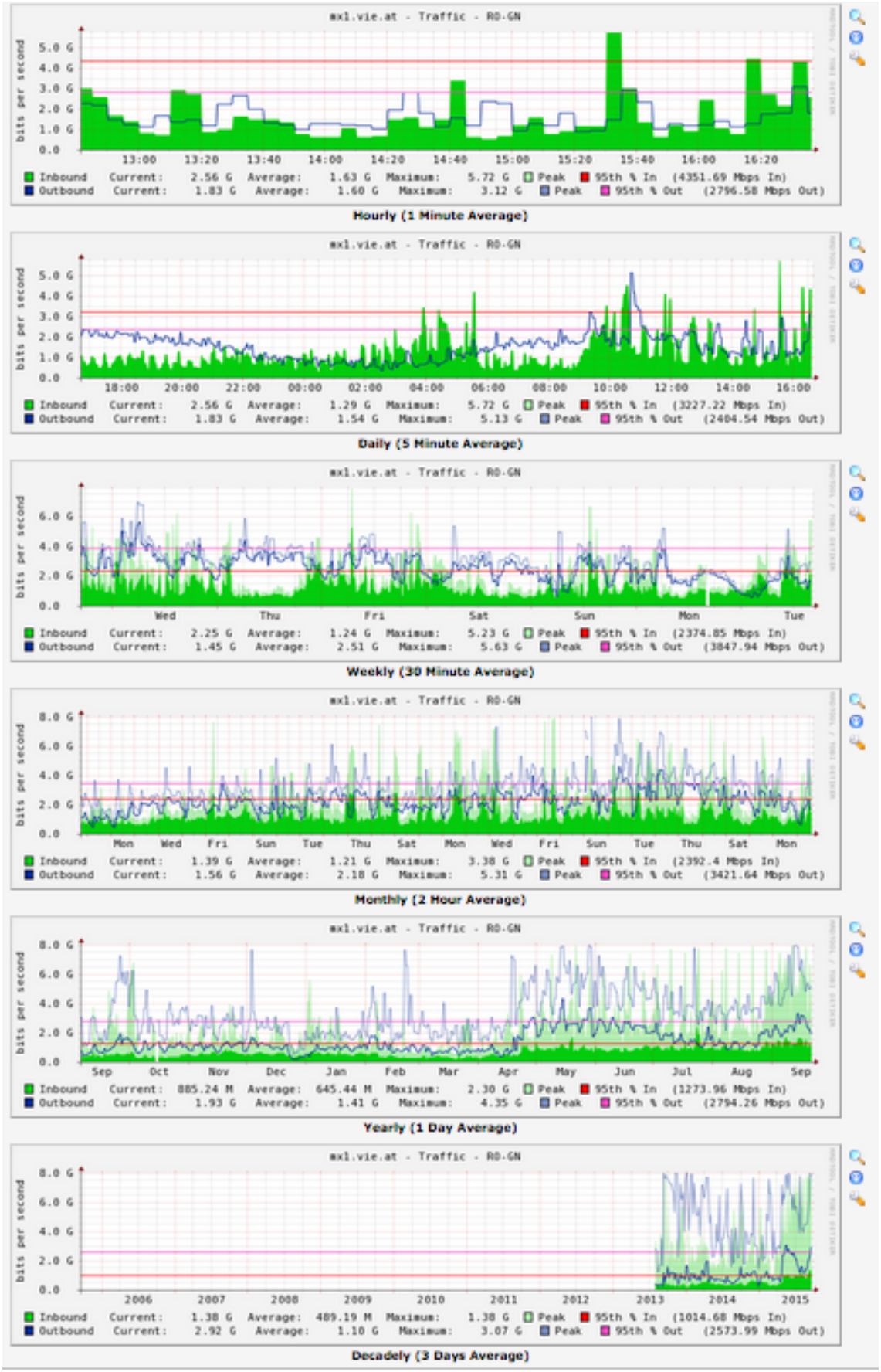


Figura 59 – Visualización del tráfico del enlace RO - GN de la red GÉANT con Cacti.

Cacti incluye gran parte de las funcionalidades necesarias para cualquier corporación:

- Gestión de archivos RRA y datos RRD mediante una interfaz completa en formato web.
- Scripts y comandos externos y soporte a SNMP.
- Configuración y generación completa de gráficos RRD.
- Sencilla configuración de interfaz SNMP de visualización de datos.
- Gestión granular de derechos de usuario.

Cabe decir que esta herramienta supuso toda una innovación en aquel entonces al ser capaz de dar soporte al más novedoso sistema de almacenamiento de datos de la fecha (RRDTool) y a la vez presentarse en la plataforma con mejor perspectiva de futuro (la plataforma web).

VENTAJAS

- Es modular.
- Utiliza la plataforma web, de forma que es fácilmente distribuible.
- Permite distintos perfiles y permisos de usuario.
- Es mucho más que un analizador de rendimiento (Alertas de umbrales, monitorización de fuentes de datos específicas en tiempo real, creación y envío de informes programados, análisis y registro de distintos sistemas, integración con otros tipos de software, hardware de seguimiento de red, copias de seguridad de configuraciones de red, etc.).

INCONVENIENTES

- La integración de las visualizaciones en plataforma web las vuelca mediante etiquetado “<img>” resultando prácticamente imposible la interactividad del usuario sobre éstas y limitándola única y exclusivamente a la generación de “tooltips” donde volcar nuevo contenido utilizando el mismo etiquetado.
- La única forma de actualizar el contenido de la imagen es mediante la substitución de la imagen bien con una consulta AJAX y su posterior reemplazo en Javascript o con a recarga de la página completa. En cualquiera de los casos anteriormente descritos resulta imposible para el usuario obtener una sensación de continuidad en la percepción de esta actualización de datos.

X.2.6. PHP Network Visualization

Es una herramienta de visualización de datos de red en forma de mapas. Utiliza datos externos provistos con plugins existentes, dando soporte a RRDTools, MRTG, archivos tsv, SNMP, fping , scripts externos y datos Cacti. Permite su integración con herramientas modernas (Cacti, Cricket, Zenoss, MRTG, Routers2, Munin, etc.

Se encuentra en desarrollo en la actualidad en su versión 0.97c, a fecha de Abril de 2013.

X.2.7. Otras

Las posibilidades, hoy en día, son infinitas y existe una guerra abierta en el sector para conseguir la más amplia penetración de la herramienta propia en el mercado.

Así, encontramos un conjunto de herramientas modernas y multifuncionales como por ejemplo Hyperic HQ, Nagios, Capsa Free, Nmap-ZenMap, EtherApe, InterMapper, etc… cada una de ellas con distintas funcionalidades.